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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20-3СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Ликов І.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2023 року

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

Ликову Іллі Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 13-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності*

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 кібербезпеки в промислову експлуатацію.
6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2023 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	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Ликов І.В.
(прізвище та ініціали)

АНОТАЦІЯ

Ликов І.В. Програмне забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

Метою розробки є програмне забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

Результат роботи – програмна реалізація системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

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 архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Visual C#.

Ключові слова: кібербезпека, мережеві дата-центри, перешкодостійке зберігання інформації

ABSTRACT

Lykov I.V. Network data center cybersecurity system software with tamper-resistant information storage to ensure integrity. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of network data centers with tamper-resistant storage of information to ensure integrity.

The purpose of the development is software for the cyber security system of network data centers with tamper-resistant storage of information to ensure integrity.

The result of the work is the software implementation of the cyber security system of network data centers with tamper-proof storage of information to ensure integrity.

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 tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Visual C# environment.

Keywords: cyber security, network data centers, interference-resistant information storage

ЗМІСТ

					ВКРБ-125.23.0040.00.00.ПЗ		
Вим	Арк.	№ докум.	Підп.	Дата			
Розроб.		Ликов І.В.			Програмне забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності	Літ.	Аркуш 1 / Аркушів 80
Перев.		Смірнов С.А.				Б	
Н.контр.		Гермак В.С.				ЦНТУ КБ-20-3СК	
Затв.		Смірнов О.А.					

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АВІ	–	адміністратора віртуальної інфраструктури
АІБ	–	адміністратор інформаційної безпеки
ВІ	–	віртуальна інфраструктура
ЗЗІ	–	засоби захисту інформації
ПД	–	персональні дані
ПЗ	–	програмне забезпечення
ЦЗОД	–	центр зберігання й обробки даних
ЦОД	–	центр обробки даних
DLP	–	захист від витоків даних
IPS	–	системи запобігання вторгнень
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

ВСТУП

Актуальність теми. Дата-центр, або центр (зберігання й) обробки даних (ЦОД/ЦЗОД) – це спеціалізований будинок для розміщення (хостингу) серверного й комунікаційного встаткування й підключення абонентів до каналів мережі Інтернет.

Дата-центр виконує функції обробки, зберігання й поширення інформації, як правило, в інтересах корпоративних клієнтів – він орієнтований на рішення бізнес-завдань шляхом надання інформаційних послуг. Консолідація обчислювальних ресурсів і засобів зберігання даних у ЦОД дозволяє скоротити сукупну вартість володіння IT-інфраструктурою за рахунок можливості ефективного використання технічних засобів, наприклад, перерозподілу навантажень, а також за рахунок скорочення витрат на адміністрування.

Дата-центри звичайно розташовані в межах або в безпосередній близькості від вузла зв'язку або точки присутності якого-небудь одного або декількох операторів зв'язку. Якість і пропускна здатність каналів впливають на рівень надаваних послуг, оскільки основним критерієм оцінки якості роботи будь-якого дата-центра є час доступності сервера (аптайм).

Призначення будь-якого дата-центра полягає в забезпеченні набору сервісів для доступу до них інших систем і додатків дата-центра або кінцевих користувачів. Число цих додатків може ранжируватися від невеликої кількості до сотень і тисяч. Гетерогенні додатки дата-центра можуть генерувати величезний трафік, що буде лягати на мережні пристрої. Особливо важким завданням у таких умовах є впровадження межмережних екранів, якому найчастіше не приділяють належну увагу або відмовляються від них взагалі.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

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

– Огляд існуючих систем мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

– Дослідження системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

– Програмна реалізація системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Призначенням системи, яка розробляється у результаті виконання бакалаврського проектування, є розробка програмного забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності

Типовий дата-центр складається з:

– інформаційної інфраструктури, що включає в себе серверне встаткування й забезпечуваючу основні функції дата-центра – обробку й зберігання інформації;

– телекомунікаційної інфраструктури, що забезпечує взаємозв'язок елементів дата-центра, а також передачу даних між дата-центром і користувачами;

– інженерної інфраструктури, що забезпечує нормальне функціонування основних систем дати-центра.

Інженерна інфраструктура містить у собі: кондиціювання для підтримки температури й рівня вологості в заданих параметрах; безперебійне електропостачання для автономної роботи дата-центра у випадках відключення центральних джерел електроенергії; охоронно-пожежну сигналізацію й система газового пожежогасіння; системи віддаленого IP контролю, керування живленням і контролю доступом.

Деякі дата-центри пропонують клієнтам додаткові послуги з використання встаткування по автоматичному відході від різних видів атак. Команди кваліфікованих фахівців цілодобово роблять моніторинг всіх серверів. Необхідно відзначити, що послуги дата-центрів сильно відрізняються в ціні й кількості послуг. Для забезпечення схоронності даних використовуються резервні системи

копіювання. Для запобігання крадіжки даних, у дата-центрах використовуються різні системи обмеження фізичного доступу, системи відеоспостереження. У корпоративні (відомчих) дата-центрах звичайно зосереджена більшість серверів відповідної організації. Устаткування кріпиться в спеціалізованих стійках і шафах. Як правило, у дата-центр приймають для розміщення лише встаткування в стієчному виконанні, тобто в корпусах стандартних розмірів, пристосованих для кріплення в стійку. Комп'ютери в корпусах настільного виконання незручні для дата-центрів і розміщаються в них рідко.

Класифікація

По відповідності вимогам стандартів

У ряді країн є стандарти на встаткування приміщень дата-центрів, що дозволяють об'єктивно оцінити здатність дата-центра забезпечити той або інший рівень сервісу. Наприклад, у США прийнятий американський (ANSI) стандарт TIA-942, що несе в собі рекомендації зі створення дата-центрів, і ділячий дата-центри на типи по ступені надійності. Хоча в Україні поки немає такого стандарту, дата-центри оснащуються відповідно до вимог для споруджень зв'язку, а також орієнтуються на вимоги TIA-942 і використовують додаткову документацію Uptime Institute і ДСТ серії 34.

Фактично, TIA-942 сприймається в усьому світі як єдиний стандарт для дата-центрів, однак слід зазначити що він досить давно не обновлявся і його досить складно застосувати в умовах України. У той же час зараз активно розвивається стандарт BICSI 002 2010 Data Center Design and Implementation Best Practices, що з'явився в 2010 і оновлений в 2011. За словами творців стандарту "стандарт BICSI 002 2010, у створенні якого брали участь більше 150 експертів, доповнює існуючі стандарти TIA, CENELEC і ISO/IEC для центрів обробки даних". Кожний зі стандартів, як правило має свою внутрішню класифікацію дата-центрів по сукупності їхніх параметрів.

Приведемо цю класифікацію.

За розміром

Великі дата-центри мають свій будинок, спеціально сконструйований для забезпечення найкращих умов розміщення. Звичайно вони мають свої канали зв'язку, до яких підключають сервери.

Середні дата-центри звичайно орендують площадку певного розміру й канали певної ширини (ширина каналу виміряється його пропускною здатністю в Мбіт/с).

Малі дата-центри розміщаються в малопристосованих приміщеннях. Часто ними використовується встаткування поганої якості, а також надається самий мінімум послуг.

За надійністю

Основний показник роботи ЦОД – відказостійкість; також важлива вартість експлуатації, показники енергоспоживання й регулювання температурного режиму.

Наприклад, стандарт TIA-942 припускає чотири рівні надійності дата-центрів:

– Tier 1 (N) – відмови встаткування або проведення ремонтних робіт приводять до зупинки роботи всього дата-центра; у дата-центрі відсутня фальшпідлога, резервні джерела електропостачання й джерела безперебійного живлення; інженерна інфраструктура не зарезервована.

– Tier 2 (N+1) – є невеликий рівень резервування; у дата-центрі є фальшпідлога й резервні джерела електропостачання, однак проведення ремонтних робіт також викликає зупинку роботи дати-центра.

– Tier 3 (N+1) – є можливість проведення ремонтних робіт (включаючи заміну компонентів системи, додавання й видалення встаткування, що вийшло з ладу) без зупинки роботи дата-центра; інженерні системи однократно зарезервовані, є кілька каналів розподілу електроживлення й охолодження, однак постійно активний тільки один з них.

– Tier 4 (2(N+1)) – є можливість проведення будь-яких робіт без зупинки роботи дата-центра; інженерні системи дворазово зарезервовані, тобто продубльовані як основна, так і додаткова системи (наприклад, безперебійне живлення представлене двома ІБП, що працюють за схемою N+1).

За призначеням

Дата-центри по виду використання підрозділяють на корпоративні, призначені для обслуговування конкретної компанії, і комерційні (аутсорсингові), що надають послуги всім бажаючої. Також розділяють провайдерозалежні й провайдеронезалежні дати-центри. Перші служать для забезпечення діяльності телекомунікаційних операторів, другі можуть використовуватися різними компаніями відповідно до їх потреб.

1.2 Область застосування

Послуги дата-центрів:

– Віртуальний хостинг. Великі дата-центри звичайно не надають подібну масову послугу через необхідність забезпечення технічно-консультаційної підтримки.

– Віртуальний сервер. Надання гарантованої й лімітованої частини сервера (частини всіх ресурсів). Важлива особливість даного виду хостингу – поділ сервера на кілька віртуальних незалежних серверів, реалізованих програмним способом.

– Виділений сервер. Дата-центр надає клієнтові в оренду сервер у різній конфігурації. Великі дата-центри в основному спеціалізуються саме на подібних типах послуг.

– Colocation. Розміщення сервера клієнта на площадці дата-центра за певну плату. Вартість залежить від енергоспоживання й тепловиділення розташовуваного встаткування, пропускної здатності каналу, що підключається до устаткування, передачі даних, а також розміру й ваги стійки.

– Оренда телекомунікаційних стійок. Передача клієнтові стійок для монтажу власного або клієнтського встаткування. Формально це окремий випадок colocation, але з основною відмінністю в тім, що орендарі в основному юридичні особи.

– Виділена зона (Dedicated area). У деяких випадках власники дата-центра виділяють частина технологічних площ для спеціальних клієнтів, як правило, фінансових компаній, що мають строгі внутрішні норми безпеки. У цьому випадку дата-центр надає якусь виділену зону, забезпечену каналами зв'язку, електропостачанням, холодопостачанням і системами безпеки, а клієнт сам створює свій дата-центр усередині цього простору.

Мережна інфраструктура

Комунікації дата-центра найчастіше базуються на мережах з використанням протоколу IP. Дата-центр містить трохи роутерів і свитчів, які управляють трафіком між серверами й «зовнішнім миром». Для надійності дата-центр іноді підключений до інтернету за допомогою безлічі різних зовнішніх каналів від різних провайдерів.

Деякі сервери в дата-центрі служать для роботи базових мережних і інтранет-служб, які використовуються усередині організації: поштові сервера, прокси-сервера, DNS-сервера й т.п.

Мережний рівень безпеки підтримують межмережні екрани, VPN-шлюзи, IDS-системи й т.д. Також використовуються системи моніторингу трафіку й деяких додатків.

«Треш-дата-центри»

В останні роки на тлі глобальної кризи, одержали розвиток проекти максимально бюджетних «треш-дата-центрів». Як правило, це великий склад або ангар, мінімально пристосований для розміщення стійок, що по класифікації TIA-942 не завжди відповідає навіть рівню Tier 1. Клієнт забезпечується негарантованим електроживленням, підключенням до Інтернет і місцем для стійки. Якщо клієнт бажає мати гарантоване електроживлення, то повинен

поставити власний ІБП у стійку. Охолодження встаткування відбувається за рахунок природного повітрообміну в приміщенні.

Достоїнством даної послуги для клієнта є вкрай низька ціна, що становить до 20-30 % від вартості розміщення сервера в дата-центрах рівня Tier 2 і Tier 3. Також у випадку даного типу дата-центрів можна говорити про деяку екологічність, оскільки немає витрат на охолодження приміщення й устаткування (у випадку ж «звичайного» дата-центра на це витрачається до 30 % від величини його загального енергоспоживання).

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

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

vGate R2

vGate R2 – це продукт, що займає окрему, що відносно недавно з'явилася нішу серед програмних засобів захисту інформації (ЗЗІ). Його покликання – забезпечувати безпеку віртуальних інфраструктур, що працюють під керуванням систем VMware Infrastructure 3 і VMware vSphere версій 4 і 4.1 у дата-центрах. Варто помітити, що лінійка продуктів VMware vSphere є спадкоємцем лінійки продуктів VMware Infrastructure.

В vGate R2 реалізований поділ ролей адміністратора інформаційної безпеки (АІБ) і адміністратора віртуальної інфраструктури (АВІ), що, по задуму розроблювачів, забезпечує додатковий рівень захисту. Фактично, АІБ відповідає за безпеку віртуальної інфраструктури (ВІ), а АВІ – за її функціонування. Також у термінах vGate ці категорії користувачів часто називаються «Адміністратор» і «Користувачі» відповідно.

У числі об'єктів захисту vGate перебувають як засоби керування ВІ, так і віртуальні машини, які використовуються в рамках цієї інфраструктури.

До засобів керування ВІ ставляться:

– ESX-сервери, призначені для хостингу й запуску віртуальних машин (ВМ).

– Сервери vCenter, призначені для централізованого керування віртуальною інфраструктурою.

– Засоби, призначені для обслуговування інфраструктури (VMware Consolidated Backup, VMware Update Manager і ін.).

– Сторонні засоби моніторингу й керування інфраструктурою.

Серед засобів захисту засобами керування BI vGate пропонує наступні:

– Автентифікація суб'єктів доступу.

– Дискреційне розмежування доступу до засобів керування BI (при цьому трафік між клієнтами й захищаємими объектами, підписується (підраховуються контрольні суми), чим попереджуються атаки типу Man in the Middle (MITM)).

– Обмеження повноважень АІБ по керуванню BI (АІБ займається забезпеченням безпеки BI, але має обмежений вплив на керування BI як такий).

– Контроль дій АВІ (з погляду забезпечення безпеки).

– Блокування доступу з веб у захищаємий периметр.

– Керовані парольні політики (можливість створення правил, що забезпечують необхідну надійність паролів і їхню ротацію).

– Повноважне керування доступом до конфіденційних ресурсів (більше гнучке керування доступом, чим дискреційне розмежування за допомогою завдання міток безпеки).

– Блокування мережного трафіку з боку віртуальних машин до засобів керування BI.

– Фільтрація мережного трафіку до серверів vCenter усередині мережі адміністрування (для цього в vGate пропонується спеціальний компонент, призначений для захисту vCenter).

– Забезпечення довіреного програмного середовища ESX-сервера.

– Контроль монтування зовнішніх пристроїв до ESX-сервера.

Серед механізмів захисту віртуальних машин пропонуються наступні:

– Контроль цілісності ВМ і докладний аудит змін у конфігураційному файлі.

– Твердження зміни конфігурації ВМ в АІБ.

– Заборона створення знімків стану ВМ (т.зв. snapshot).

– Заборона клонування ВМ.

– Очищення пам'яті ВМ (гарантують відсутність залишкової інформації про оброблювані дані в пам'яті ВМ).

– Гарантоване видалення без можливості відновлення залишкової інформації в системі зберігання даних ESX-сервера при видаленні ВМ (атака типу «складання сміття»).

– Контроль пристроїв, що підключаються.

– Обмеження завантаження файлів із ВМ.

– Обмеження доступу до консолі ВМ (у якій відображається інтерфейс операційної системи, установленої у ВМ і додатків, що працюють під керуванням цієї ОС під час роботи із ВМ).

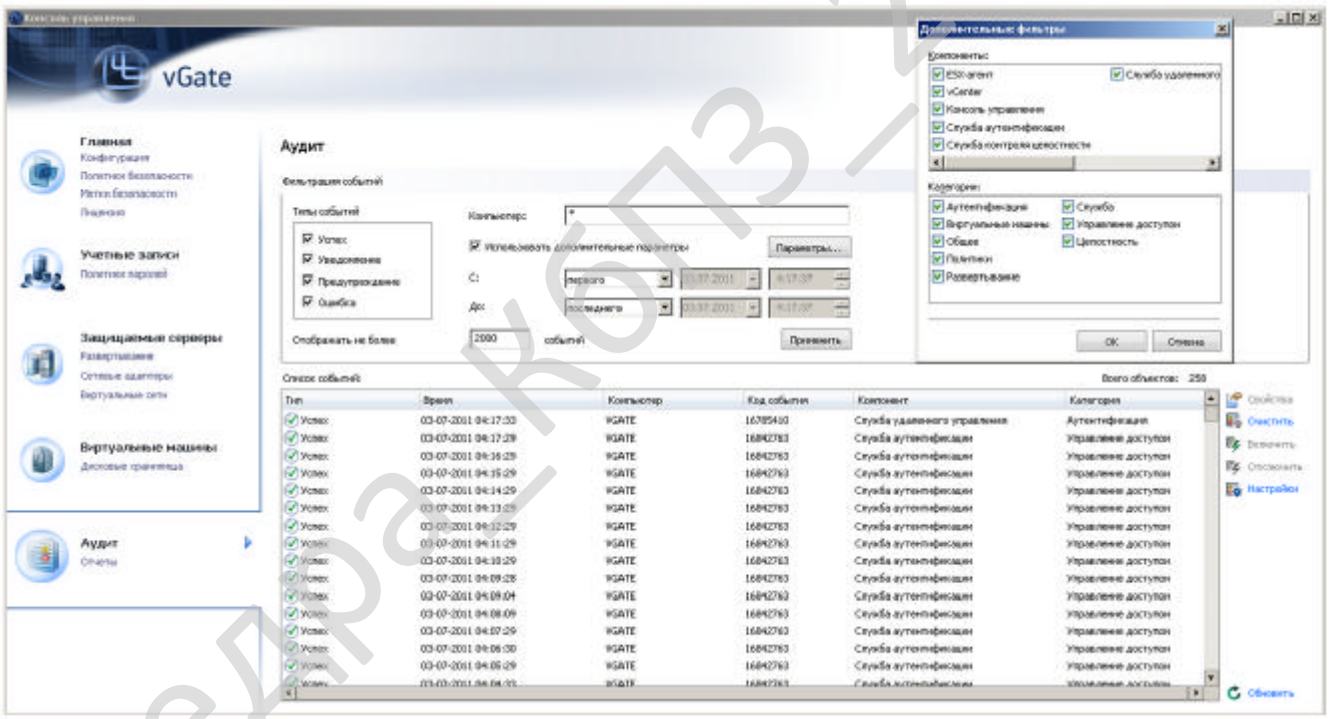

Рисунок 2.1 – Інтерфейс користувача vGate R2

При цьому, перераховані механізми захисту ВМ спрямовані на захист від несанкціонованого доступу до об'єктів що захищається ВІ. Повноцінний захист ВМ як кінцевих точок у загальноприйнятому змісті в vGate R2 відсутній. Для організації такого захисту виробник vGate R2 пропонує використовувати

додаткові продукти, такі як засіб захисту інформації від несанкціонованого доступу Secret Net і міжмережний екран TrustAccess того ж виробника.

Сервер авторизації vGate виконує наступні функції:

– Централізоване керування ЗЗІ vGate.

– Автентифікація користувачів і комп'ютерів.

– Розмежування доступу до засобів керування ВІ.

– Реєстрація подій безпеки.

– Зберігання даних (облікової інформації, журналів аудита й конфігурації vGate).

– Реплікація даних (при наявності резервного сервера).

Резервний сервер авторизації виконує наступні функції:

– Зберігання настроювань і списку користувачів.

– Реплікація даних.

– Можливість заміни основного сервера у випадку його можливого виходу з ладу.

Модуль захисту ESX-сервера виконує наступні функції:

– Контроль цілісності й довірене завантаження ВМ.

– Контроль цілісності модулів і настроювань vGate.

– Реєстрація подій безпеки.

– Захист від несанкціонованого доступу усередині мережі адміністрування.

– Контроль монтування пристроїв.

– Забезпечення довіреного програмного середовища.

Компонент захисту vCenter установлюється безпосередньо на сервер, де встановлений що захищається vCenter Server і виконує наступні функції:

– Захист від несанкціонованого доступу усередині мережі адміністрування.

– Керування фільтрацією вхідного трафіку.

vGate Client установлюється на робочі місця АІБ і АВІ в зовнішньому периметрі мережі й виконує наступні функції:

– Ідентифікація й автентифікація користувача.

– Ідентифікація й автентифікація комп'ютера.

– Контроль цілісності компонентів агента автентифікації.

– Вибір рівня сесії при роботі з конфіденційними ресурсами (рівень доступу можна змінювати під час роботи).

– Реєстрація подій безпеки.

Сервер звітів vGate може бути встановлений на будь-який комп'ютер зовнішнього периметра мережі, включаючи комп'ютер, на якому працює vGate Server. Сервер звітів служить для формування звітів про стан параметрів безпеки ВІ, що відбулися подіях і внесених у конфігурацію змінах.

vGate R2 є представником окремої, щодо недавно, що з'явилася ніші, продуктів, присвячених інформаційної безпеки, – захисту об'єктів віртуальних інфраструктур підприємств від несанкціонованого доступу. При цьому відомі й інші рішення для захисту віртуальних інфраструктур, наприклад, сімейство продуктів VMware vShield, Trend Micro Deep Security, IBM Virtual Server Protection. Необхідність використання продукту vGate R2 незаперечна, він дозволяє доповнити систему захисту, здійснювану цими рішеннями засобами захисту від несанкціонованого доступу й контролю виконання політик інформаційної безпеки з урахуванням специфіки віртуального середовища. Спільне використання всіх перерахованих засобів захисту дозволить компаніям одержати впевненість у захисті своєї віртуальної інфраструктури від компрометації.

Розгортання й настроювання vGate у локальній мережі підприємства не повинна викликати значних труднощів при дотриманні системних вимог і обмежень, викладених у документації. Настроювання політик і параметрів доступу до захищаємих об’єктів, віртуальної інфраструктури досить гнучкі.

Відповідні настроювання й майстри в інтерфейсі консолі керування vGate розташовані досить зручно.

Серед недоліків vGate, які авторові огляду здалися важливими, можна відзначити наступні:

– Неможливість установити vGate Server на 64-бітні ОС.

– Відсутність підтримки протоколу IPv6.

– Необхідність використання платної версії MS SQL Server 2005 для установки сервера звітів vGate (правда, вартість цього додаткового продукту, мабуть, становить невелику частку від вартості vGate).

– При установці агента авторизації vGate Client у систему встановлюється додаткова мережна служба з назвою z0NdisHelper Filter, наявність якого може викликати проблеми з мережними підключеннями. Зокрема, в автора огляду відмовився працювати Інтернет по протоколі ETTH доти, поки галочка, що відповідає зазначеній службі не була знята у властивостях мережного адаптера.

– Відсутність можливості установки компонента захисту vCenter і керування їм з консолі керування vGate (хоча для відповідного компонента захисту ESX-серверів така можливість є).

Проте, зрозуміло, що розроблювачі таких продуктів, як vGate R2 постійно самі зіштовхуються з «качанами» у системній і іншій додатковій вимогах, які вже існують у таких складних системах як VMware vSphere, і додають (часто вимушено) свої вимоги.

Trend Micro Deep Security

Компанія Trend Micro є одним з основних гравців на ринку забезпечення інформаційної безпеки. Її продукти використовуються у всіх сегментах ринку – починаючи від домашніх користувачів і закінчуючи великим бізнесом і постачальниками послуг.

Останнім часом один з напрямків роботи компанії – це створення систем захисту «хмарних» обчислювальних систем, і, зокрема, віртуальних центрів обробки даних (ЦОД) або окремих віртуальних серверів. Актуальність даного напрямку зв'язана зі стрімким зростанням популярності технологій віртуалізації,

які використовуються для побудови хмарних структур, публічних (public) або часток корпоративних (private) хмар.

Популярність віртуалізації зв'язана також з істотною економією витрат на покупку й обслуговування встаткування за рахунок консолідації ресурсів.

Захист віртуальних ЦОД по своїй природі відрізняється від захисту звичайних кінцевих точок. При використанні віртуалізації виникають нові погрози, які не були актуальні раніше для звичайних систем:

– Виключені («сплячі») віртуальні машини не захищені. Часте включення й вимикання віртуальних машин приводить до того, що досить складно забезпечувати й підтримувати їхня безпека. Наприклад, якщо віртуальна машина виключена при розгортанні або відновленні антивірусу (або інших модулів безпеки), те при переході в активний стан вона буде незахищена.

– Конфлікт ресурсів. Роботи, які пов'язані з інтенсивним використанням ресурсів на окремих віртуальних машинах, наприклад, антивірусні перевірки або відновлення антивірусних баз, можуть привести до великого завантаження системи в цілому. Якщо на кожній віртуальній машині в хості починають виконуватися такі однотипні операції, то зростає кількість використовуваних системних ресурсів і, як наслідок, уповільнюється робота прикладних додатків. Таким чином, кількість ресурсів і величина часових затримок лінійно зростає зі збільшенням кількості віртуальних машин на хості.

– Ріст кількості віртуальних машин. При збільшенні кількості серверів і віртуальних машин виникають проблеми з конфігуруванням, настроюванням і супроводом систем безпеки. У результаті з'являються «вузькі» місця в забезпеченні безпеки й збільшується складність керування всією системою.

– Трафік між віртуальними машинами в межах одного гіпервізора не контролюється звичайними засобами мережного моніторингу. У результаті можливі атаки між віртуальними машинами на одному хості.

– Безпека існує у відриві від даних. Захищаються окремі хости й віртуальні машини, а не самі дані, які постійно переміщаються. У результаті,

якщо віртуальні машини перемістилися між хостами, те потрібна перенастроювання, що вимагає значних часових витрат.

Для великих підприємств також актуальною проблемою є відповідність систем захисту діючим галузевим стандартам в області безпеки. Наприклад, PCI DSS і FISMA.

Таким чином, використання систем безпеки, неадаптованих для віртуального середовища, не тільки не знижує описані вище ризики, але й приводить до нераціонального використання системних ресурсів.

З урахуванням нових погроз практичніше використовувати захист безпосередньо на рівні гіпервізора. Остання повинна містити не тільки антивірусний захист, брандмауера й IPS (Intrusion Prevention System, система виявлення вторгнень), що вміють автоматично переконфігуруватися слідом за змінами віртуального середовища, але й контроль цілісності для віртуальних машин, що захищаються. Для рішення цих завдань компанія Trend Micro випустила продукт Trend Micro Deep Security, який ми й розглянемо в даному огляді.

Архітектура й компоненти Trend Micro Deep Security

Deep Security Manager

Deep Security Manager – це система централізованого керування, що дозволяє адміністраторам створювати профілі безпеки й застосовувати їх до що захищається Deep Security Agent або Deep Security Virtual Appliance серверам. За допомогою централізованої консолі Deep Security Manager можна управляти політиками безпеки, відслідковувати попередження про погрози й автоматично вживати дії для їхнього усунення, розсилати відновлення для системи безпеки по серверах, а також створювати звіти про свою роботу з різним рівнем деталізації. Адміністрування здійснюється на основі розподілу ролей. При інсталяції продукту створюється запис головного адміністратора (master administrator), що може створювати облікові записи з різним рівнем повноважень для роботи в рамках усього комплексу.

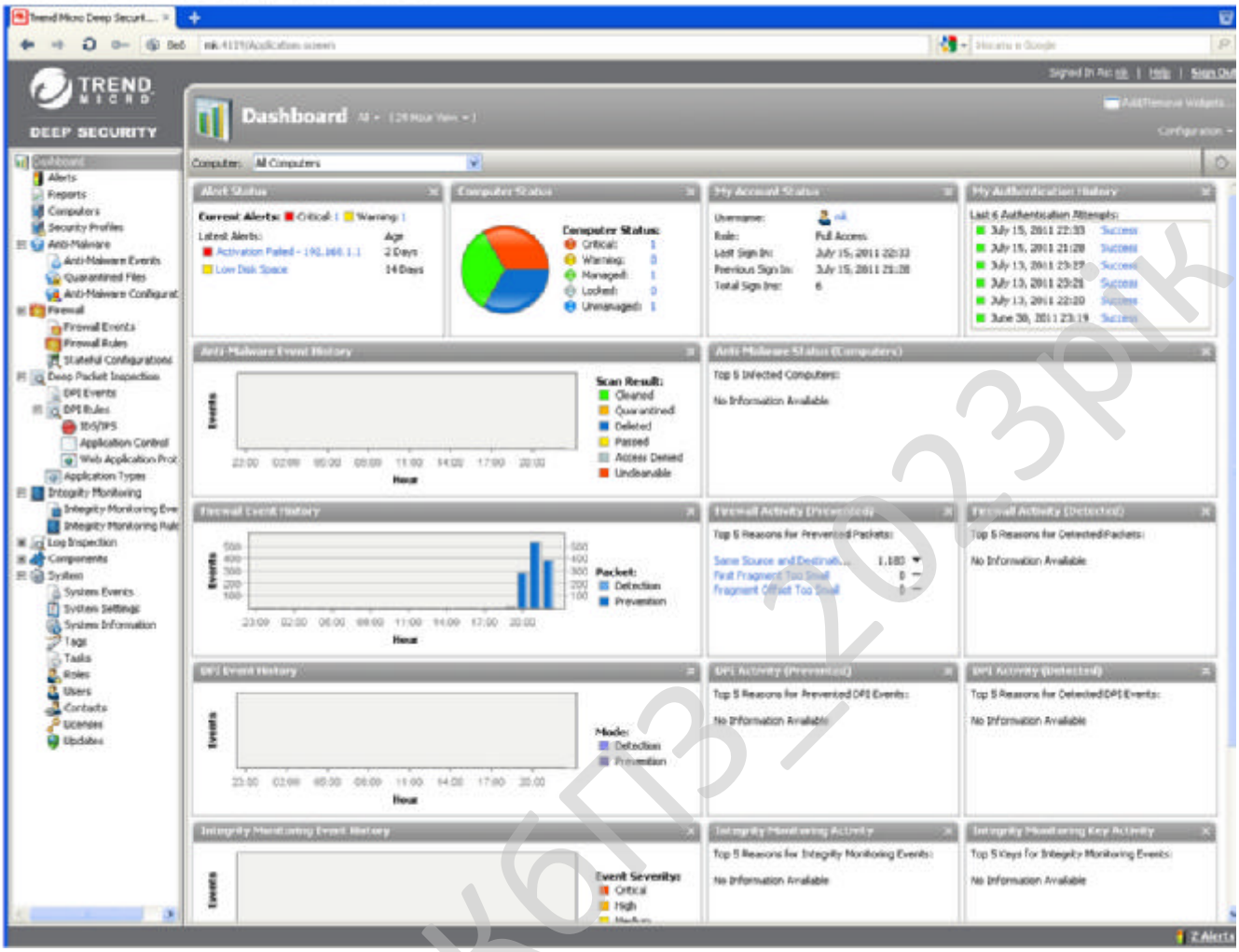

Рисунок 2.2 – Інтерфейс користувача Trend Micro Deep Security

З метою забезпечення відказостійкості й масштабованості Deep Security Manager може бути дубльований на декількох машинах. Дублювання припускає кілька паралельних інсталяцій менеджера, які повинні спільно використовувати одну базу даних. У цьому випадку всі екземпляри менеджера рівноправні, і адміністратор може використовувати веб-консоль кожної із цих інсталяцій. При такому підході збій одного з вузлів не приводить до втрати інформації або функціональності.

Deep Security Agent

Даний компонент установлюється на захищаємий сервер, або віртуальну машину для застосування централізованих політик безпеки. У нього входить система виявлення й запобігання вторгнень (IDS/IPS), що виконує функції захисту веб-додатків і керування додатками, брандмауер, модулі для контролю цілісності файлової системи й реєстру, а також перевірки записів журналів роботи операційної системи й додатків, що ставляться до безпеки.

Віртуальний пристрій Deep Security Virtual Appliance

Використовується для реалізації політик безпеки на віртуальних машинах VMware vSphere або ESX /ESXi. Deep Security Virtual Appliance забезпечує функціональність IDS/IPS, виконує функції захисту веб-додатків, керування додатками, брандмауера, а також містить безагентський антивірусний модуль. Існує можливість координації з агентом Deep Security Agent для контролю цілісності критичних областей операційної системи й прикладних додатків, а також перевірки записів системних журналів, що ставляться до безпеки. Дозволяє проводити перевірку віртуальних машин на хості без установки агента на кожну віртуальну систему.

Security Center

Клієнтський портал забезпечує доступ до відновлень, які можуть завантажуватися автоматично або по запиті, а також за допомогою Deep Security Manager і встановлюватися на все сервера в організації.

Smart Protection Network

«Хмарна» інфраструктура компанії Trend Micro, що дозволяє в режимі реального часу оцінювати репутацію веб-сайтів, поштових повідомлень і файлів. Smart Protection Network дозволяє блокувати нові погрози (шкідливі файли, спам, фішінг-атаки, заражені веб-сайти, DDoS-атаки, уразливості, погрози втрати даних, а також комбіновані атаки), не прибігаючи до класичного відновлення сигнатур, а одержуючи інформацію прямо з «хмари» практично миттєво. В останній версії – Trend Micro Deep Security реалізована можливість інтеграції

Deep Security Manager з Smart Protection Network для підтримки правил, у контексті антивірусного файлового захисту, у всіх захищаємих системах, в актуальному стані.

Принцип роботи Trend Micro Deep Security

Trend Micro Deep Security забезпечує комплексний захист фізичних, віртуальних і хмарних серверів. Особливістю даної системи є те, що вона дозволяє захищати віртуальну інфраструктуру на базі продуктів фірми VMWare, використовуючи API для прив'язки до гіпервізору.

Trend Micro Deep Security може бути розгорнута у формі програмної платформи, віртуального пристрою або у вигляді гібридного рішення. При цьому може забезпечуватися захист як окремих віртуальних машин, так і їхньої сукупності, розгорнутої на одному хості.

Залежно від розв'язуваного завдання на конкретну фізичну або віртуальну машину встановлюється один із двох видів додатків – Deep Security Agent або Deep Security Virtual Appliance. Якщо коштують завдання моніторингу цілісності й/або збору подій з журналів систем і додатків на фізичних або віртуальних машинах, а також існує можливість того, що ці системи можуть «переїхати» у незахищене середовище, то встановлюється Deep Security Agent. Для захисту всіх установлених на хості віртуальних машин від мережних погроз і проникнення вірусів на рівні гіпервізора використовується Deep Security Virtual Appliance, що встановлюється на виділеній віртуальній машині (Security Virtual Machine) на хості. Таким чином, при захисті віртуальних середовищ можна взагалі обійтися без установки агентів на хостах.

Можливість захисту віртуальних машин на рівні гіпервізора забезпечується тісною інтеграцією Trend Micro Deep Security з API VMware vShield Endpoint і VMware vShield Manager. Це дозволяє здійснювати швидке розгортання на хостах VMware vSphere і забезпечувати прозорий захист віртуальних машин VMware vSphere.

У цілому, Trend Micro Deep Security заслуговує позитивної оцінки. Концепція продукту розроблена з урахуванням сучасних погроз безпеки віртуальних середовищ і, незважаючи на вже тривале існування, усе ще претендує на інноваційність на ринку. Сам продукт містить велику кількість компонентів забезпечення безпеки, робота з менеджером досить проста й зрозуміла. Якихось помітних недоліків виявлено не було, як мінуси можна позначити кілька дріб'язків і побажань. Наприклад, не вистачає інструментів для аналізу вихідного трафіку на предмет витоків конфіденційних даних.

Переваги**:**

– Використання окремої виділеної віртуальної машини Deep Security Virtual Appliance для забезпечення безпеки. Даний підхід дозволяє не ставити клієнт у кожну віртуальну машину й забезпечувати безпека всіх віртуальних машин на хості на рівні гіпервізора. У результаті вирішуються специфічні для віртуальних середовищ проблеми безпеки, оптимізується споживання ресурсів системи й зменшується час перевірок. Окремо варто згадати про кешувані результатів тестування, що збільшує продуктивність сканування й мінімізує трафік між віртуальними машинами.

– Великий набір компонентів безпеки. Trend Micro Deep Security містить у собі систему виявлення й запобігання вторгнень, брандмауер, засоби контролю цілісності файлів і додатків, захисту веб-додатків, перевірки журналів, а також захисту від шкідливих програм без використання агента.

– Системи, що захищаються. Trend Micro Deep Security може встановлюватися як на фізичні машини, так і у віртуальні середовища. Підтримується захист великої кількості операційних систем (різні версії Windows, Solaris, Linux, AIX, HP-UX).

– Тісна інтеграція з VMware vSphere і ESX/ESXi. Це дозволяє Trend Micro Deep Security взаємодіяти з віртуальним середовищем на рівні драйверів і дає доступ для контролю передачі даних на рівні гіпервізора. При інтеграції з VMware vSphere забезпечується захист віртуальних машин без використання агента.

– Користувальницький інтерфейс. Незважаючи на велику кількість параметрів працювати в системі досить зручно. Це пов'язано з декількома факторами: інтерфейс не перевантажений зайвими деталями, функції програми жорстко структуровані, алгоритми роботи з різними компонентами захисту побудовані по загальному принципі.

– Підтримка. Trend Micro Deep Security інтегрована з інфраструктурою Smart Protection Network, що забезпечує постійне автоматичне відновлення сигнатур і правил з єдиного центра.

– Документація до Trend Micro Deep Security виконана на високому рівні. Докладно описується послідовності установки під різні операційні системи й робота кожного компонента. Окремо варто сказати, що документація написана технічною мовою й не містить різних рекламних і маркетингових вставок.

Недоліки**:**

– Користувальницький інтерфейс і документація неукраїнізовані. З урахуванням великої кількості параметрів, що налаштовуються, це трохи незручно.

– Антивірусний модуль є тільки в Deep Security Virtual Appliance, в агенті, якому можна ставити на фізичні машини, даний модуль відсутній. Однак у версії 8.0, вихід якої запланований на 4-й квартал 2012 року, Trend Micro планує додати дану функціональність для агентів під Windows і Linux.

– Користувальницький інтерфейс. Є ряд дріб'язків, які можна поліпшити. Іконки різних інструментів маленькі й не завжди відбивають ту функціональність, до якої надається доступ. Деякі параметри в інформаційному вікні не мають назв.

– «Спеціальні» ризики. У випадку відмови Deep Security Virtual Appliance, наприклад, у випадку атаки на нього, всі віртуальні машини, що захищаються, стануть уразливі. Також в експертному співтоваристві є сумніву в роботі віртуального патчинга, тому що його використання може приводити до відмов у роботі захисту.

IBM Security (Proventia) Network Intrusion Prevention System

IBM Security Network Intrusion Prevention System (раніше ISS Proventia Network Intrusion Prevention System) призначений для блокування мережних атак і аудита роботи мережі, а також збільшення пропускної здатності за рахунок видалення "паразитного" і непродуктивного трафіку. Завдяки запатентованій технології аналізу протоколів дане рішення IBM забезпечує проактивний захист – своєчасний захист корпоративної мережі від широкого спектра погроз. Проактивність захисту заснована на цілодобовому відстеженні погроз у центрі забезпечення безпеки й власних досліджень і пошуках уразливостей аналітиками й розроблювачами групи IBM X-Force.

IBM Security Network IPS як і всі продукти серії IBM Security ґрунтується на технології Protocol Analysis Module (PAM, модуль аналізу протоколів), розробленої й підтримуваним дослідниками й розроблювачами команди **IBM X-Force**.

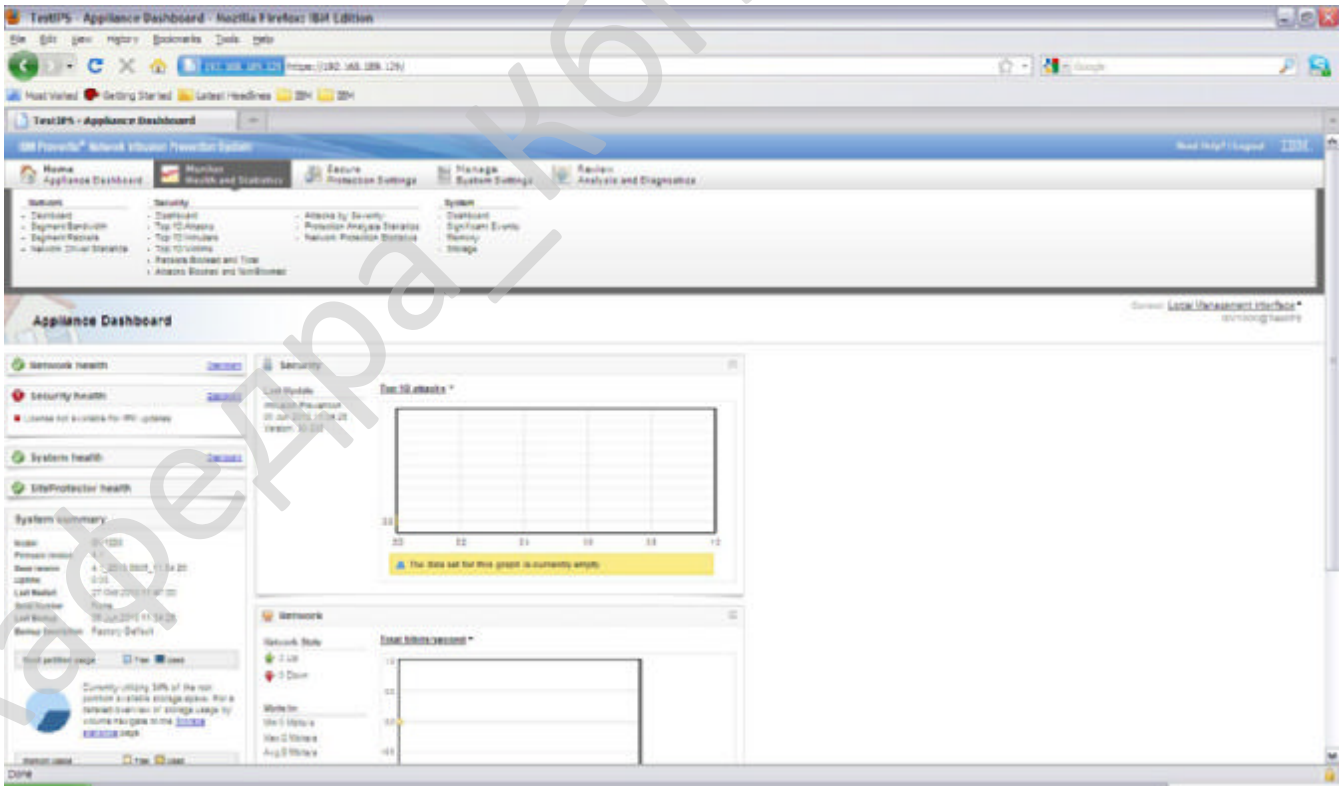

Рисунок 2.3 – Інтерфейс користувача IBM Security (Proventia) Network Intrusion Prevention System

Модуль аналізу протоколів складається з технологій:

– Віртуальний патч – закриває можливість використання уразливостей у ПЗ поза залежністю від того чи встановлено на системі відповідне відновлення від виробника й забезпечує можливість відмовитися від "пожежних" мір при поширенні відновлень для ПЗ й ОС. Критичні системи як і раніше повинні обновлятися якнайшвидше, однак у цьому випадку ви можете витрачати досить часу на тестування відповідних відновлень. Інші продукти IBM Security, такі як, наприклад, IBM Security Network Enterprise Scanner, можуть також допомогти швидше ідентифікувати найбільш уразливе ПЗ з метою більше ефективного керування процесом установки відновлень.

– Захист додатків на ПК – захищає кінцевих користувачів від атак, спрямованих на додатки, використовувані в щоденній роботі, такі як Microsoft Office, Adobe PDF, різні мультимедіа функції й веб-браузери.

– Захист веб-додатків – забезпечує захист веб-серверів і веб-додатків від атак на рівні програмного коду, таких як SQL Injection, XSS (Cross-Site Scripting), PHP file-includes, CSRF (Cross-Site Request Forgery) і ін.

– Виявлення й блокування погроз – виявляє й блокує цілий клас атак (експлойтів), спрямованих на використання тієї або іншого уразливості.

– Безпека даних – контролює й виявляє спроби передачі персональні дані (ПД) і інші типи конфіденційної інформації. Також забезпечує можливість досліджувати переміщення тої або іншої інформації усередині й поза корпоративною мережею для більше ранньої оцінки потенційного ризику в майбутньому.

– Контроль додатків – дозволяє управляти правами доступу для недозволених додатків і при необхідності блокувати їхній доступ до внутрішніх або зовнішніх ресурсів. Серед підтримуваних типів додатків, такі як Active елементи, P2P-додатка, засобу обміну миттєвими повідомленнями (ICQ, Skype і ін.) і спроби туннелювання трафіку.

Рішення IBM Security Network Intrusion Prevention System поставляється у вигляді:

– Програмно-апаратного комплексу в складі моделей: Proventia GX 4004-V2-200 (до 200 Мбіт/с оброблюваної пропускної здатності), Proventia GX 4004-V2 (до 800 Мбіт/с), GX 5008-V2 (до 1,5 Гбіт/с), GX 5108-V2 (до 2,5 Гбіт/с), GX 5208-V2 (до 4 Гбіт/с) і GX6116 (до 8 Гбіт/с).

– Програмного комплексу для високопродуктивної платформи Crossbeam.

– Програмного комплексу для платформи віртуалізації VMware у складі опцій: GV200 (до 200 Мбіт/с оброблюваної пропускної здатності на апаратній конфігурації, що рекомендується) і GV1000 (до 700 Мбіт/с на апаратній конфігурації, що рекомендується).

Додаткові опції IBM Security Network Intrusion Prevention System включають:

1. Програмно-апаратний комплекс IBM Security Network Active Bypass – гарантує відказостійкость системи IBM Security Network IPS на базі моніторингу й автоматичного перенапрямку трафіку у випадку апаратної або програмної відмови для моделей без убудованого модуля обходу (для GX 5008-V2 і вище).

2. Програмно-апаратний комплекс IBM Security Network Controller забезпечує можливість використання з'єднань 10 Gb (Gigabit Ethernet) (для GX6116 і GX 5208-V2).

Основні характерні риси рішень компанії IBM по мережній безпеці серії IBM Security (Proventia) Network Intrusion Prevention System**:**

– Широкий модельний ряд програмних або програмно-апаратних комплексів дозволяє організувати надійний захист мереж передачі даних у компанії будь-якого розміру й будь-якого рівня складності мережної інфраструктури, надаючи можливість аналізу трафіку від 200 Мбіт/секунду до 40 Гбіт/секунду без затримок і переривань.

– Рішення засноване на проактивних технологіях захисту, які дозволяють ефективно блокувати атаки в самому широкому спектрі й запобігати спроби

проникнення в мережу, а також ефективно знижувати ризики дій внутрішнього персоналу.

– Функціональні можливості продукту набагато ширше класичного подання про системи запобігання вторгнень (IPS), включаючи також такі функції як міжмережне екранування, захист від витоків даних (DLP), захист веб-додатків і СУБД, контроль доступу додатків до мережі й ін.

– Широкі засоби звітності й архівації даних надають великі можливості для аудита й аналізу поточного рівня забезпечення інформаційної безпеки.

– Забезпечує безпека для всієї мережі в цілому, дозволяючи нівелювати ризики відсутності відновлень на окремих ПК або серверах.

До мінусів продукту можна віднести відсутність локалізації й документації українською мовою. Крім того, у продукті поки відсутня можливість аналізу SSL трафіку, для цього буде потрібно рішення від сторонніх компаній.

Побудова системи інформаційної безпеки підприємства на базі рішень від Cisco Systems

Безпека кінцевого хоста – Cisco Security Agent

Рішення Cisco Security Agent (CSA) являє собою систему забезпечення безпеки кінцевого хоста й у зв'язуванні з іншими системами дозволяє вирішувати більше складні й широкі завдання.

CSA забезпечує захист серверних систем і настільних комп'ютерів. Можливості Cisco Security Agent перевищують можливості типових рішень для захисту кінцевих вузлів, поєднуючи в рамках одного програмного засобу передові функції захисту від цілеспрямованих атак, шпигунських програм, програм для схованого віддаленого керування, антивірусний захист, а також захист від витоків інформації й багатьох інших типів порушення безпеки комп'ютера.

Cisco Security Agent являє собою систему, що використовує агентські додатки для застосування настроєних на центральному сервері політик інформаційної безпеки.

CSA поєднує в собі захист від « zero-day» атак, антивірус ClamAV, міжмережний екран, модуль захисту файлів і додатків, модуль «недовірених» додатків і інші функції.

Cisco Security Agent надає цілий ряд коштовних можливостей, серед яких можна виділити наступні:

– контроль відповідності стану об'єктів мережі вимогам політики безпеки;

– превентивний захист від цілеспрямованих атак;

– контроль USB, CD-ROM, PCMCIA і т.д.;

– створення замкнутого програмного середовища;

– здатність виявлення й ізоляції шкідливих програм для схованого віддаленого керування;

– розвинені функції запобігання вторгнення на вузли мережі, персонального міжмережного екрана й захисту від зовсім нових атак;

– контроль витоку інформації;

– контроль і запобігання завантаження з несанкціонованих носіїв;

– оптимізація використання смуги пропущення Wi-Fi;

– забезпечення доступності критично важливих додатків « клієнт-сервер» і можливості здійснення транзакцій;

– маркірування мережного трафіку;

– інтеграцію із системами запобігання вторгнень (Cisco IPS);

– інтеграція із системою контролю доступу до мережі (Cisco NAC);

– інтеграція із системою керування безпекою (Cisco MARS).

Контроль доступу до мережі – Cisco Network Admission Control (NAC)

Cisco NAC Appliance (колишній Cisco Clean Access) – це рішення, призначене для автоматичного виявлення, ізолювання й лікування інфікованих,

уразливих або не відповідній політиці безпеки вузлів, що здійснюють провідний або бездротової доступ до корпоративних ресурсів.

Будучи одним з компонентів технології Network Admission Control, Clean Access виконаний або у вигляді мережного модуля для маршрутизаторів Cisco ISR (для мереж з кількістю контрольованих пристроїв менш 100), або у вигляді окремого пристрою.

Рисунок 2.4 – Інтерфейс користувача Cisco NAC

Основними можливостями рішення Cisco NAC є:

– незалежність від виробника мережного встаткування (у режимі in-band);

– інтеграція з Kerberos, LDAP, RADIUS, Active Directory, S/Ident і іншими методами автентифікації;

– підтримка ОС Windows (включаючи Vista), MacOS, Linux, Xbox, PlayStation 2, КПК, принтерів, IP-телефонів і т.д.;

– підтримка антивірусів CA, F-Secure, Eset, «Лабораторії Касперського», McAfee, Panda, Dr;Web, Sophos, Symantec, TrendMicro і інших засобів захисту комп'ютера (усього 250 виробників);

– приміщення невідповідного вузла в карантин шляхом застосування списків контролю доступу ACL або VLAN;

– створення "білого" списку вузлів для прискорення їхнього доступу до ресурсів мережі;

– автоматична установка відсутніх відновлень, нових версій засобів захисту або актуалізація застарілих антивірусних баз;

– централізоване web-керування;

– підтримка російської мови;

– проведення прозорого аудита.

Cisco Security Monitoring, Analysis and Response System (MARS)

Cisco MARS являє собою програмно-апаратне рішення в серверному виконанні. Програмне забезпечення системи базується на операційній системі Linux (ядро 2.6). Основним компонентом системи є база даних Oracle, що використовується для зберігання інформації.

Cisco MARS має можливість збору інформації з різних пристроїв по протоколах Syslog, SNMP, NetFlow, а також має можливість приймати системні перелоги-файли.

MARS підтримує встаткування різних вендорів таких як Cisco, IBM, Check Point, Nokia, Symantec, McAfee, Netscape і інших.

Логіка роботи системи Cisco MARS базується на запитах до бази даних. Можна вибирати інформацію й уточнювати її по IP-Адресі джерела, IP-Адресі приймача, портам, типам подій, пристроям, по ключових словах і так далі.

На базі запитів базуються певні правила, які групуються в системі. У базі Cisco MARS містить більше 2000 правил. Можна створювати свої правила, тим самим гнучко адаптувати систему до конкретних видів передбачуваних погроз.

Після збереження правила й виявлення інформації, що задовольняє даному правилу – формується інцидент.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Програмне забезпечення написане мовою C#. Ця мова обрана виходячи з наступних міркувань. C# – строго типізована об'єктно-орієнтована мова, призначена для розробки різноманітних безпечних і потужних додатків, виконуваних у середовищі .NET Framework. Мовою 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# 2008 є розширений редактор коду, конструктори зі зручним користувальницьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку додатків мовою C# версії 3.0 і .NET Framework версії 3.5.

Синтаксис C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C++ або Java з легкістю взнають синтаксис із фігурними дужками, характерний для мови C#. Розроблювачі, що знають кожну із цих мов, як правило, зможуть домогтися ефективної роботи з мовою C# за дуже короткий час. Синтаксис 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. 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 C# 3.0 вираження LINQ (Language-Integrated Query) роблять строго-типізований запит першокласною конструкцією мови.

Як об'єктно-орієнтована мова, C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод `Main` – точку входу додатка – інкапсуються 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 C# структура схожа на полегшений клас: це тип, що розподіляється по стопках, що реалізує інтерфейси, але не підтримуюче спадкування.

На додаток до основних описаних об’єктно-орієнтованих принципів, мова 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 C# можна використовувати процес, що називається "Interop". Процес Interop дозволяє програмам на C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C++. Мова C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення.

Процес побудови 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 C# може бути визначене будь-яке число класів, структур, інтерфейсів і подій.

Архітектура платформи .NET Framework

Програма мовою 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 C#, компілюється в проміжну мову (IL) у відповідності зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення й рядки, зберігаються на диску 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.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

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 кібербезпеки 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 кібербезпеки 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) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Центри обробки даних, на даний момент, класифікуються по 4-рем рівням надійності – tier 1, tier 2, tier 3, tier 4. Чим вище рівень надійності дата-центра, тим більше надійний ЦОД.

Пропонований інструмент для оцінки надійності дата-центра у вигляді певних параметрів і вимог до інженерних систем, дозволяє оцінити й визначити рівень надійності ЦОД не тільки споживачам, які планую розмістити своє встаткування або скористатися послугами й сервісами центра обробки даних, але й інвесторам, які ухвалюють рішення щодо вкладення свого капіталу в будівництво дати-центра. Також цей підхід у вигляді певних вимог дозволяє оцінити підприємствам, які планують побудувати для рішення своїх завдань свій особистий некомерційний ЦОД із заданим рівнем tier, що їм потрібно для забезпечення роботи бізнес-процесів. Наприклад, якомусь підприємству цілком допустимо простий і кілька мінут у день, а значить і не потрібна супернадійність (наприклад, рівень tier 4), а деяким комерційним підприємствам, простий у кілька мінут на місяць обійдеться серйозними фінансовими втратами й упущеною вигодою, тому їм необхідно побудувати ЦОД з рівнем tier 4 .

Стандарт TIA EIA 942

Розробки Uptime Institute в області визначення рівнів надійності з його дозволу «перекочували» у північноамериканський стандарт TIA EIA 942, прийнятий в 2005 році. Для кожного з виділених рівнів надійності в стандарті TIA EIA 942 приводиться детальний опис, вимоги й рекомендації до наступних систем і елементів: архітектурних рішень, електропостачання, охолодження, безпеки, протипожежній системи, структурованої кабельної системи, системи

кабеледротів, телекомунікацій. Наприклад, у стандарті приводиться опис вимог і рекомендацій з мінімальної висоти фальшпідлоги для певного рівня надійності.

У стандарті при описі вимоги до кількості ресурсів використовується буква «N» (скорочення від слова need) і найпростіші математичні формули з операціями додавання й множення. Операція додавання позначає збільшення потреби на одну умовну одиницю (наприклад, N+1 означає, що необхідно мати запас ресурсів в одну одиницю), а операція множення збільшення потреб у кілька разів (наприклад, N*2 означає повне дублювання потреб).

1-ий рівень надійності ЦОД – tier 1

Базовий рівень надійності ЦОД. Цей рівень застосовувався для дата-центрів в 60-і й 70-і роки минулого сторіччя. Помилки й відмови в роботі систем і встаткування на цьому рівні приводять до збоїв у роботі всього ЦОД. Також робота центра обробки даних переривається для проведення профілактичних і ремонтних робіт. У ЦОД може не бути фальшпідлоги, резервних джерел електропостачання й джерел безперебійного живлення (ІБП).

Інженерна інфраструктура створена тільки для задоволення поточних потреб, тобто без резервування й надлишкових ресурсів (забезпечення потреб виражається у вигляді букви «N»).

Час простою за рік – 28,8 годин.

Коефіцієнт відказостійкості 99,671%.

2-ий рівень надійності ЦОД– tier 2

Дата-центри на 2-ом рівні мають невеликий рівень резервування працездатності систем і мають невеликі надлишкові ресурси в інженерних системах датацентра. Але однаково піддані перебоям через планові й непланові відмови роботи встаткування в дата-центрах. Для цього рівня необхідно мати фальшпідлогу, резервні джерела електропостачання ЦОД. Проведення технічних і ремонтних робіт зажадає зупинку роботи центра обробки даних.

Система не має повного резервування, однак установлені додаткові елементи в системах охолодження й енергопостачання ЦОД (забезпечення потреб виражається у вигляді формули «N+1»).

Час простою за рік – 22,0 години.

Коефіцієнт відказостійкості 99,749%.

3-ій рівень надійності ЦОД – tier 3

Дата-центр із даним рівнем надійності дозволяє провести ремонтно-профілактичні роботи без зупинки роботи ЦОД. Тобто можлива одночасно експлуатація й технічне обслуговування центра обробки даних аж до заміни компонентів системи, додавання й видалення встаткування, що вийшов з ладу. Щоб забезпечити 3-ий рівень уже необхідно для системи охолодження спроектувати й побудувати два трубопроводи, забезпечити резервними потужностями роботу всього встаткування з урахуванням виходу з ладу або профілактики системи електропостачання. Але помилки в роботі й відмови можуть викликати перебої в роботі дати-центра.

Має кілька шляхів (каналів) для розподілу електроживлення й охолодження, але лише один з них активний; має резервовані компоненти (забезпечення потреб виражається у вигляді формули «N+1»).

Час простою за рік – 1,6 години.

Коефіцієнт відказостійкості 99,982%.

4-ий рівень надійності датацентра – tier 4

Відказостійкий дата-центр із резервуванням всіх систем, що дозволяє виконати будь-які планові й позапланові роботи без переривання роботи ЦОД. На цьому рівні забезпечується надійний захист від збоїв. Щоб відповідати вимозі 4-ого рівні надійності необхідні дублювання всіх систем з обліком того, що в кожній системі і її «резервній копії» буде перебувати, як мінімум, ще один додатковий компонент, що забезпечує резервування за схемою «N+1». Тобто в дата-центрі повинне бути резервування системи на рівні «N+1» і сама система ще повинна бути, як мінімум, продубльована. Відмови можуть мати місце у

випадках ручного аварійного відключенні системи електропостачання й спрацьовування системи пожежної безпеки. На 4-му рівні навіть структурована кабельна система повинна бути повністю зарезервована.

Системи мають подвійне резервування з обліком, як мінімум, додаткового компонента. Має кілька активних шляхів розподілу навантаження й охолодження з резервними компонентами 2 (N+1), тобто 2 ІБП із надмірністю N+1 кожний (забезпечення потреб виражається у вигляді формули «2 (N+1)»).

Час простою за рік – 0,4 години.

Коефіцієнт відказостійкості 99,995%.

Не треба також забувати, що в ході експлуатації дата-центра й додавання серверів і встаткування систем зберігання даних у ЦОД при незмінній інженерній інфраструктурі базові потреби в дата-центрі виростуть і це може привести до зміни рівня надійності центра обробки даних. Тобто необхідно переглядати рівень надійності ЦОД. Але не факт, що про зміну рівня надійності повідомить власник комерційного дата-центра – адже це не в його інтересах. Поки в стандартах не розглядаються питання рівня надійності центрів обробки даних залежно від рівня експлуатації.

Швидше за все, буде прийнятий Uptime Institute 5-ий рівень надійності ЦОД tier 5 з коефіцієнтом відказостійкості виді 5-ти дев'яток 99,999%, що буде відповідати потребам сучасних підприємств.

Безпека дата-центра в умовах збільшення навантаження

Оскільки всі сервера в дата-центрі можна розділити на організаційні категорії відповідно до вимог замовників, то в ідеальному випадку, кожна логічна група хостів повинна бути безпечно обмежена від інших. Доступ між цими групами повинен бути або обмежений, або закритий відповідно до політики безпеки організації. У реальному житті це неможливо, оскільки межмережний екран повинен пропустити величезну кількість підключень до кожного сервера, обробляючи при цьому кожну сесію окремо. Саме фактор кількості нових сесій у секунду є визначальним на етапі вибору межмережного екрана для впровадження

в дата-центрі, і відстежити їхнє число дуже складно. Найчастіше можна судити лише про середнє значення швидкості нових підключень, а несподівані перегони можуть часово блокувати роботу пристрою або навіть привести до його відмови. Тому будь-який фаєрвол, впроваджуваний у дата-центрі, повинен ефективно обробляти як постійне значення швидкості встановлення нових підключень, так і перегони навантаження, у кілька разів перевищуючі середні значення.

Після обробки підключення встановлюється з'єднання й кількість сесій обслуговуються пристроєм у конкретний момент визначає ще один важливий параметр при впровадженні пристрою безпеки – кількість установлених сесій. Велика кількість серверів у дата-центрі при великій кількості підключених користувачів у результаті дає величезне число встановлених сесій. Тут можна відзначити дві важливі особливості – здатність пристрою ефективно обслуговувати вже встановлені сесії, а також, можливість обробляти нові. Неможливість створення нових з'єднань при обслуговуванні вже встановлених приведе до неприступності сервісів для деяких користувачів.

Після установки сесій ще одним фактором, що впливає на успішне проходження трафіку через фаєрвол, є загальна кількість пакетів у секунду, а також, загальна пропускна здатність для всіх сесій. Величина, характеризуюча кількість пакетів у секунду, важлива, у тому випадку, коли пристрій демонструє заявлену пропускну здатність тільки при обробці невеликої кількості пакетів великий довгі. Наприклад, число пакетів в 64-байтному потоці в 23 рази перевищує те ж число в 1500-байтному. Зміст полягає в тім, що фаєрвол може забезпечити задану пропускну здатність не для будь-якого типу трафіку. При проектуванні дата-центра завжди потрібно враховувати реальні показники продуктивності фаєрволу.

Безпека в дата-центрі вкрай важлива, тому що є ядром обробки інформації для більшості компаній. При її впровадженні дуже важливе логічне угруповання схожих сервісів, для взаємодії яких немає необхідності впроваджувати зайві міри безпеки. Наприклад, оптимально групувати веб-сервера, сервера додатків і сервера баз даних в одному логічному сегменті. Першим кроком при

забезпеченні безпеки для серверів є обмеження можливих зв'язків до необхідного мінімуму, що забезпечить захист від неавторизованого доступу. Другим – впровадження технології запобігання вторгнень (IPS), що забезпечить перевірку мережного трафіку на наявність атак. Це особливо актуально в дата-центрах, де уразливість сервісу може привести до його неприступності або втрати важливої інформації. Через небезпеку таких наслідків впровадження IPS критично важливо, але є й деякі особливості – обмеження по місцю впровадження й продуктивності. Через обмежену пропускну здатність пристрій не завжди може обробити необхідна кількість трафіку при впровадженні в певному сегменті мережі, наприклад, ядрі дата-центра, а також, неможливо вказати який трафік обробляти, а який немає. Тому, пріоритетної є селективна обробка трафіку й для визначення рівня необхідної продуктивності.

У сучасних дата-центрах для повної з’вязуємості мереж обов'язковим є використання протоколів динамічної маршрутизації. Це ж стосується й пристроїв безпеки, для яких цей функціонал не є першочерговим. Більше вузька підтримка протоколів динамічної маршрутизації може створити труднощі при впровадженні фаєрволу в інфраструктуру мережі дати-центра.

Не варто забувати про технологію віртуалізації, покликаної збільшити утилізацію ресурсів кожного сервера й зменшити їхня кількість. Але даному контексті вона являє собою додаткові перешкоди на шляху впровадження пристроїв безпеки. Використовуючи дану технологію сервера вже не підключаються до двох різних мереж , як раніше, зараз ця безліч віртуальних мереж, які необхідно обслуговувати. Цей факт ще раз підвищує роль протоколів динамічної маршрутизації. При збільшенні кількості логічних мереж може знадобитися кілька фізичних пристроїв безпеки, а доцільніше – один більше продуктивний пристрій з такими ж можливостями віртуалізації, як сервер.

Всі перераховані особливості створюють певні труднощі при плануванні й впровадженні пристроїв, покликаних забезпечити безпеку інформаційних потоків дата-центру.

3.2 Розробка структурної схеми

Для реалізації перешкодостійкого зберіганням інформації у мережних дата-центрах, застосуємо алгоритм 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 на різних носіях, у тому числі й у дата-центрах, при створенні архівів з інформацією для відновлення у випадку ушкоджень, у завадостійкому кодуванні.

Код Ріда-Соломона був винайдений в 1960 році співробітниками лабораторії Лінкольна Массачуссетского технологічного інституту Ірвином Рідом і Густавом Соломоном. Ідея 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 Месси.

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-Хоквингхема (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

*П*оліномом, 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 $c(x) = m(x)g(x)$, де ступінь $m(x)$ 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 $g(x), xg(x), x^2g(x), \ldots, x^{k-1}g(x)$ лінійно незалежні, інакше $m(x)g(x) = 0$ при ненульовому $m(x)$, що неможливо.

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

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

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

Матрицю G можна записати в символьній формі:

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

Перевірочна матриця

Для кожного кодового слова циклічного коду справедливо $c(x) = 0 \mod g(x)$. Тому перевірочну матрицю можна записати як $H = \begin{bmatrix} 1 & x & x^2 & \ldots & x^{n-2} & x^{n-1} \end{bmatrix} \mod g(x)$.

Тоді:

$$\overline{c}H^T = \sum_{i=0}^{n-1} c_i x^i \bmod g(x). \qquad (3.2)$$

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 $GF(q)$, корінням якого є d-1 ступенів, що йдуть підряд, $\alpha^{l_0}, \alpha^{l_0+1}, \ldots, \alpha^{i_0+d-1}$ елемента α, є поліномом, що породжує, коду над *полем* GF*(*q) $g(x) = (x-\alpha^{l_0})(x-\alpha^{l_0+1})\ldots(x-\alpha^{l_0+d-1})$:, 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 $l_i <= mt_i - (m - 1)$, 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 $l + (m - 1) <= mt$.

Практична реалізація

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

При несистематичному кодуванні інформаційне слово множиться на якийсь полином, 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 lnn^2.

Кодування

При операції кодування інформаційний поліном множиться на багаточлен, що породжує. Множення вихідного слова S довжини k на не приводиться полином, 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 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 перетворяться в індексну форму, потім складаються в межах байта без обліку переносу.

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

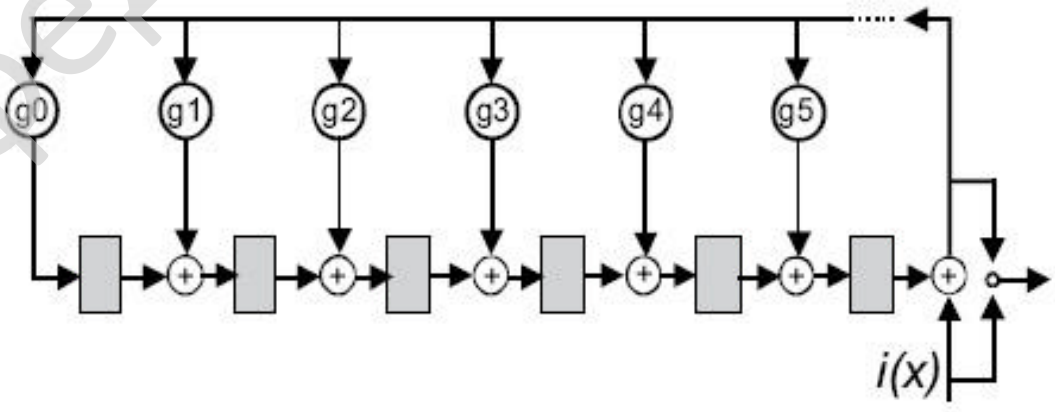

Рисунок 3.1 – Функціональна схема кодера Ріда-Соломона

При переході від однієї форми подання даних до інший доцільно використовувати таблицю істинності розміром 256 байт, що становить ємність одного EAB (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 EAB.

Для перекладу результатів у поліноміальну форму потрібно вже 16 таких таблиць, що вимагає застосування IMC 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 – EAB", мультиплексуючи константи на входах суматорів і дозволяючи роботу регістрів-накопичувачів у моменти появи відповідних даних на виходах засувок EAB.

На структурній схемі кодера (рисунок 3.2) символу «C» 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 EAB. 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 EAB, що дозволяє розмістити його в IMC EPF10K10. Після оптимізації розміщення схеми на кристалі FPGA швидкодія схеми досяглася 11,57 Мгц (частота надходження байт даних, далі – байтова частота).

При використанні IMC EPF10K20, у складі якої 6 EAB, використовуючи 4 пари "суматор – EAB", 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 $a, b, r_1 > r_2 > r_3 > r_4 > \cdots > r_n$ визначена тим, що кожне r_k це остача від ділення перед-попереднього числа на попереднє, а передостаннє ділиться на останнє нацело, тобто

$$a = bq_0 + r_1$$

$$b = r_1 q_1 + r_2$$
$$r_1 = r_2 q_2 + r_3 \quad (3.3)$$
$$\dots$$
$$r_{n-1} = r_n q_n$$

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 випливає з наступних двох тверджень:

– Нехай $a = bq + r$, тоді $(a,b) = (b,r)$.

– $(0,r) = r$. для будь-якого ненульового r.

Знаходження корня

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 Форни 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.

Код Ріда-Соломона використовується при записі й читанні в контролерах оперативної пам'яті, при архівуванні даних, запису інформації на жорсткі диски

(ECC), запису на Диск мережевого дата-центруи. 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.

Можливі помилки при читанні з диска з'являються вже на етапі виробництва диска, тому що зробити ідеальний диск при сучасних технологіях неможливо. Так само помилки можуть бути викликані подряпинами на поверхні диска, пилом і т.д. Тому при виготовленні компакт-диску, що читається, використовується система корекції CIRC (Cross Interleaved Reed Solomon Code). Ця корекція реалізована у всіх пристроях, що дозволяють зчитувати дані з CD дисків, у вигляді чипа із прошиванням firmware. Знаходження й корекція помилок заснована надмірності й перемежені (redundancy & interleaving). Надмірність приблизно 25% від вихідної інформації.

При записі на цифрові аудіокомпакт-диски (Compact Disc Digital Audio – CD-DA) використовується стандарт Red Book. Корекція помилок відбувається на двох рівнях C1 і C2.

При кодуванні на першому етапі відбувається додавання перевірочних символів до вихідних даних, на другому етапі інформація знову кодується.

Крім кодування здійснюється також перемішування (перемеження) байтів, щоб при корекції блоки помилок розпалися на окремі біти, які легше виправляються. На першому рівні виявляються й виправляються помилкові блоки довжиною один і два байти (один і два помилкових символи відповідно). Помилкові блоки довжиною три байти виявляються й передаються на наступний рівень. На другому рівні виявляються й виправляються помилкові блоки, що виникли в C2, довжиною 1 і 2 байти. Виявлення трьох помилкових символу є фатальною помилкою, не можуть бути виправлені.

Цей алгоритм кодування використовується при передачі даних по мережах WiMAX, в оптичних лініях зв'язку, у супутниковому й радіорелейному зв'язку. Метод прямої корекції помилок у минаючому трафике (Forward Error Correction,

FEC) ґрунтується на кодах Ріда-Соломона.

Структурна схема системи підвищення надійності зберігання даних у дата-центрі за допомогою кодека Ріда–Соломона зображена на рисунку 3.2.

З цієї схеми ми бачимо, що над вхідними даними, перед записом у дата-центрі, відбуваються перетворення кодеком Ріда-Соломона. Після кодування дані записуються у дата-центр. У якості носія окрім дата-центру може використовуватися любий інший носій інформації. Після цього інформація зберігається на носієві.

При зчитуванні інформації, розроблене програмне забезпечення декодує інформацію, яка зберігається на носієві, і якщо потрібно, після проведення відповідних перевірок, проводить відновлення втраченої інформації. Якщо таке відновлення неможливе, то програма видає відповідне повідомлення.

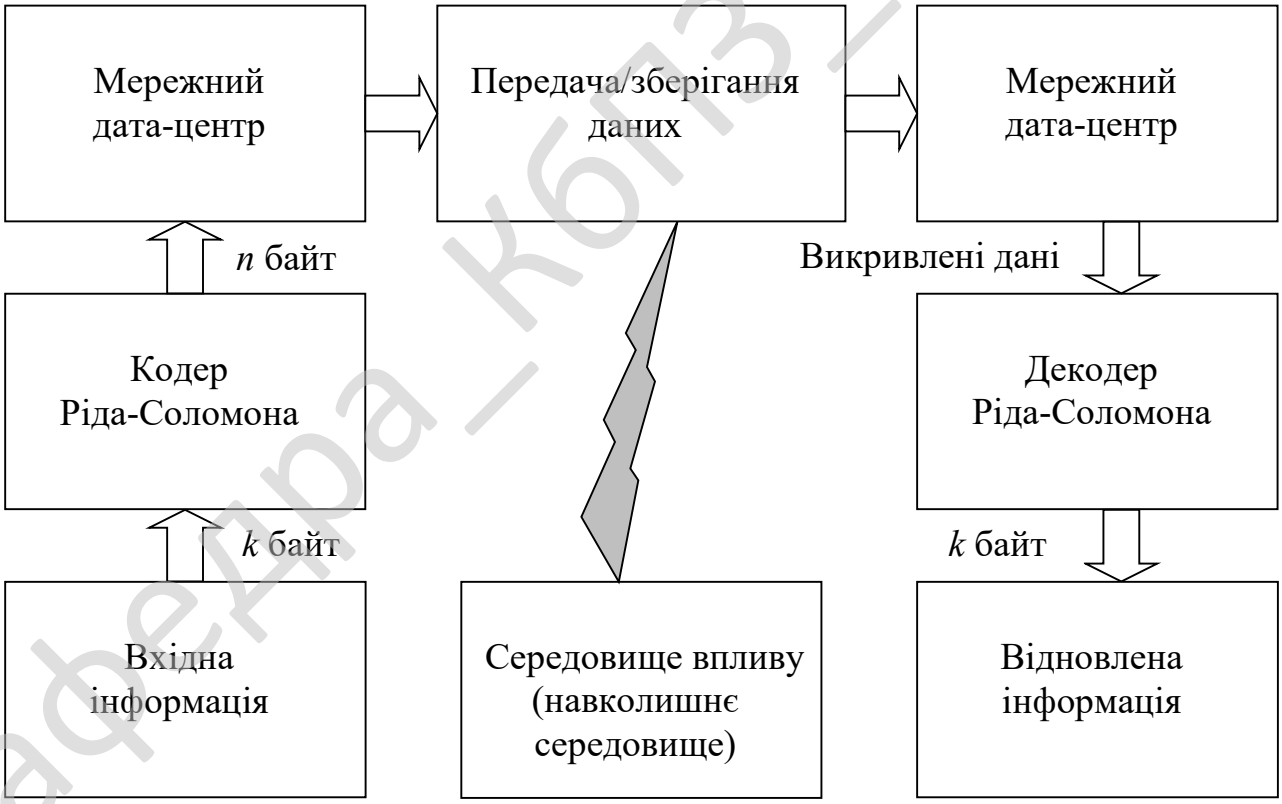

Рисунок 3.2 – Структурна схема системи

3.3 Розробка функціональної схеми

На рисунку 3.3 зображена функціональна схема системи. З неї бачимо, що розроблена система підвищення стійкості до уражень інформації яка записана на дисках складається з наступних основних функціональних блоків:

– сам носій інформації – Диск мережевого дата-центру;

– кодер Ріда-Соломона, який використовується, коли відбувається запис інформації на диск, при цьому необхідно враховувати, що об'єм інформації, яка записується на диск, повинна бути меншою, ніж об'єм диску, 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 с відповідного диску.

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

– дані, які потрібно зберегти у дата-центрі;

– поліном для кодування;

– відмовостійки коди.

Коди Ріда-Соломона базуються на спеціальному розділі математики – полях Галуа (GF) або кінцевих полях. Арифметичні дії (+, -, х, / 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.

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: $g(x) = (x-a^{i})(x-a^{i+1})\ldots(x-a^{i+2t})$, а кодове слово формується за допомогою операції:

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

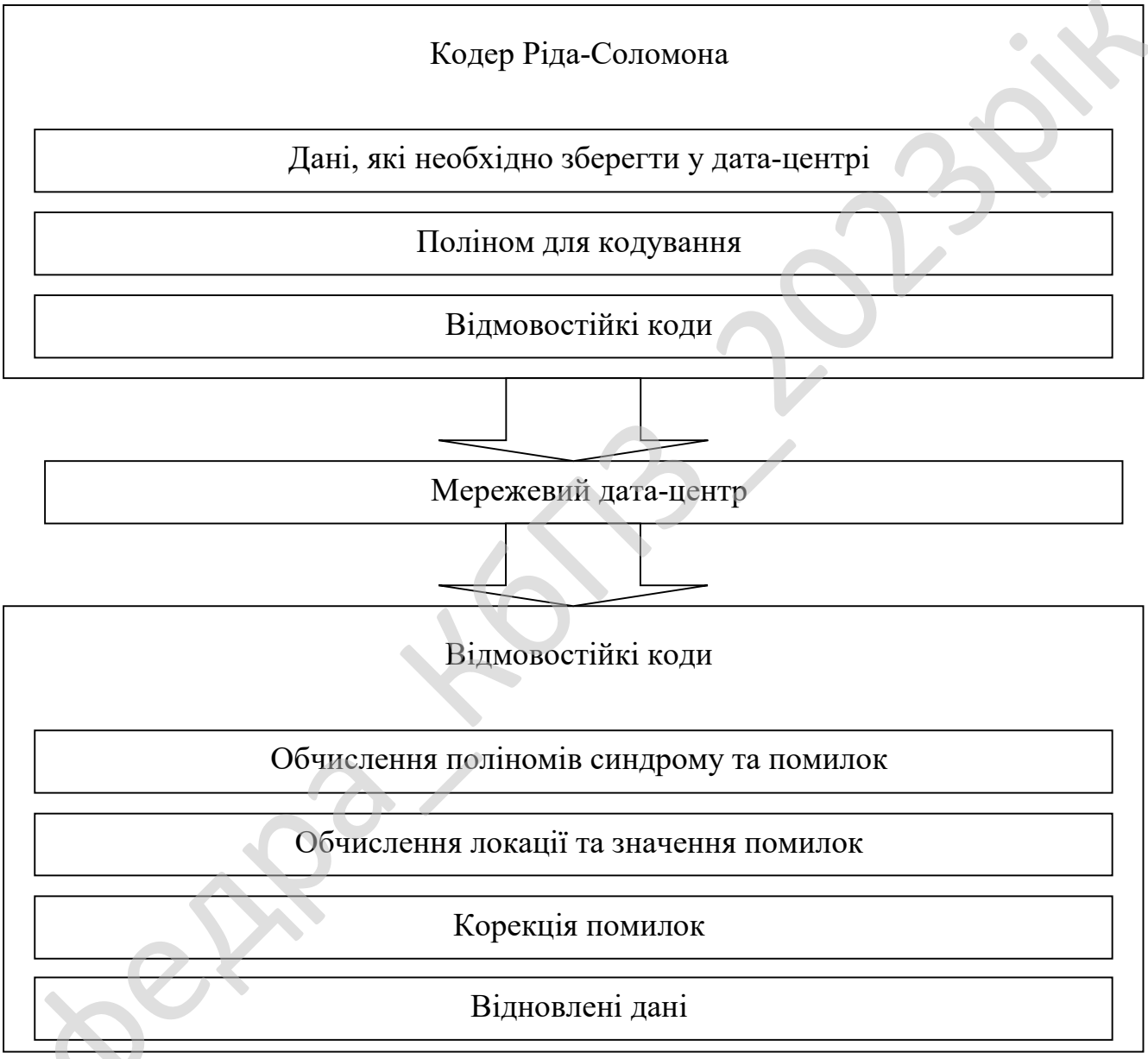

Рисунок 3.3 – Функціональна схема системи

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 коррекції помилок;

– відновлені за допомогою коді Ріда-Соломона дані.

Декодер працює наступним чином.

Введемо позначення:

– r(x) – Отримане кодове слово.

– Si – Синдроми.

– L(x) – Поліном локації помилок.

– Xi – Положення помилок.

– Yi – 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 кодових слів). Синдроми можуть бути обчислені шляхом підстановки 2t коріння утворюючого полінома g(x) в r(x).

Знаходження позицій символьних 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 символьних 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.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.4. З рисунку видно, що процеси взаємодіють наступним чином.

Спершу запускається процес виведення головного вікна програми.

Він взаємодіє з наступними процесами:

– Процес захисту даних.

– Процес вибору томів на резервних дисках.

– Процес вибору файлів.

– Процес вибору конфігурації.

Процес захисту даних взаємодіє з процесом ввімкнення/вимкнення шифрую чого фільтру, який у свою чергу, взаємодіє з процесом встановлення паролю шифрування.

Процес вибору файлів взаємодіє з наступними процесами:

– Процес вибору дисків, для запису відновленої інформації.

– Процес вибору резервних дисків.

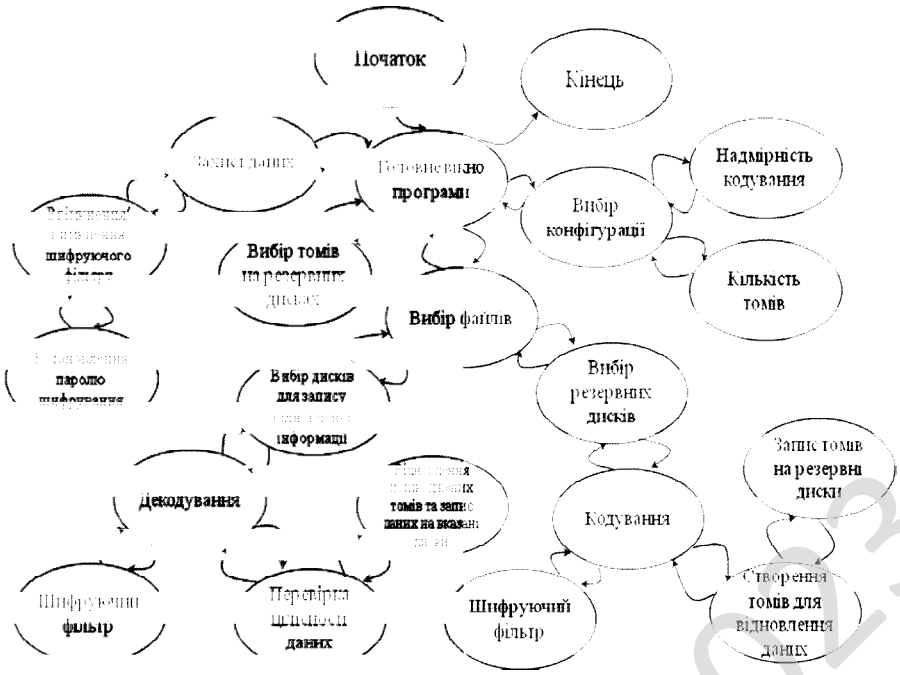

Рисунок 3.4 – Діаграма процесів

Процес вибору дисків, для запису відновленої інформації взаємодіє з процесом декодування, який, у свою чергу взаємодіє з наступними процесами:

– Процес завантаження шифруючого диску.

– Процес перевірки цілісності даних, який взаємодіє з процесом відновлення пошкоджених томів та запис даних на вказані диски.

Процес вибору резервних дисків взаємодіє з процесом кодування.

Останній процес взаємодіє з наступними процесами:

– Процес шифруючого диску.

– Процес створення томів для відновлення даних, який взаємодіє з процесом запису томів на резервний диск.

Процес вибору конфігурації взаємодіє з процесом вибору надмірності кодування та процесом вибору кількості томів.

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

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення основного вікна програми. Після цього відбувається виведення списку дисків та їх вмісту.

З цими дисками користувач може виконувати, одну з наступних дій:

– Встановити захист даних.

– Створити перешкодостійкі томи.

– Відновити інформацію.

Якщо користувач обирає встановлення захисту даних, то виконуються наступні дії:

– Ввімкнення шифруючого фільтру.

– Введення паролю шифрування.

Якщо користувач обирає створення перешкодостійких томів, то виконуються наступні дії:

– Вибір файлів для перешкодостійкого збереження.

– Вибір резервних дисків для збереження інформації.

– Вибір величини надмірності кодування.

– Вибір кількості томів для збереження інформації.

– Виклик підпрограми кодування за допомогою алгоритму Ріда-Соломона.

– Створення томів для відновлення та запис їх на резервні диски.

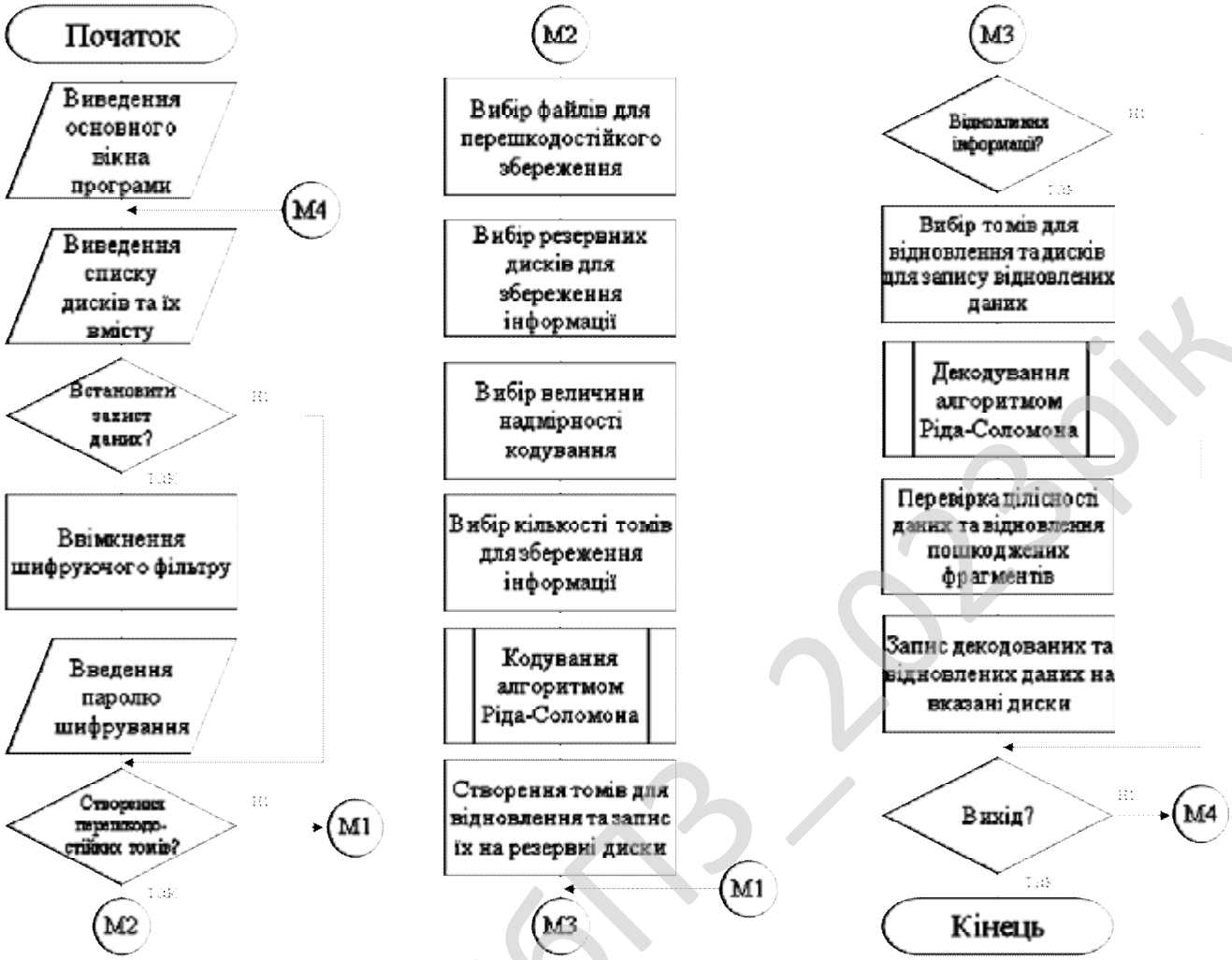

Рисунок 4.1 – Блок-схема роботи основної програми

Якщо користувач обирає відновлення інформації, то виконуються наступні дії:

– Вибір тому для відновлення та дисків для запису відновлених даних.

– Запуск підпрограми декодування алгоритмом Ріда-Соломона.

– Перевірка цілісності даних та відновлення пошкоджених фрагментів.

– Запис декодованих та відновлених даних на вказані диски.

Після виконання усіх вищеперерахованих дій, користувач обирає, працювати йому далі з програмою, або ні.

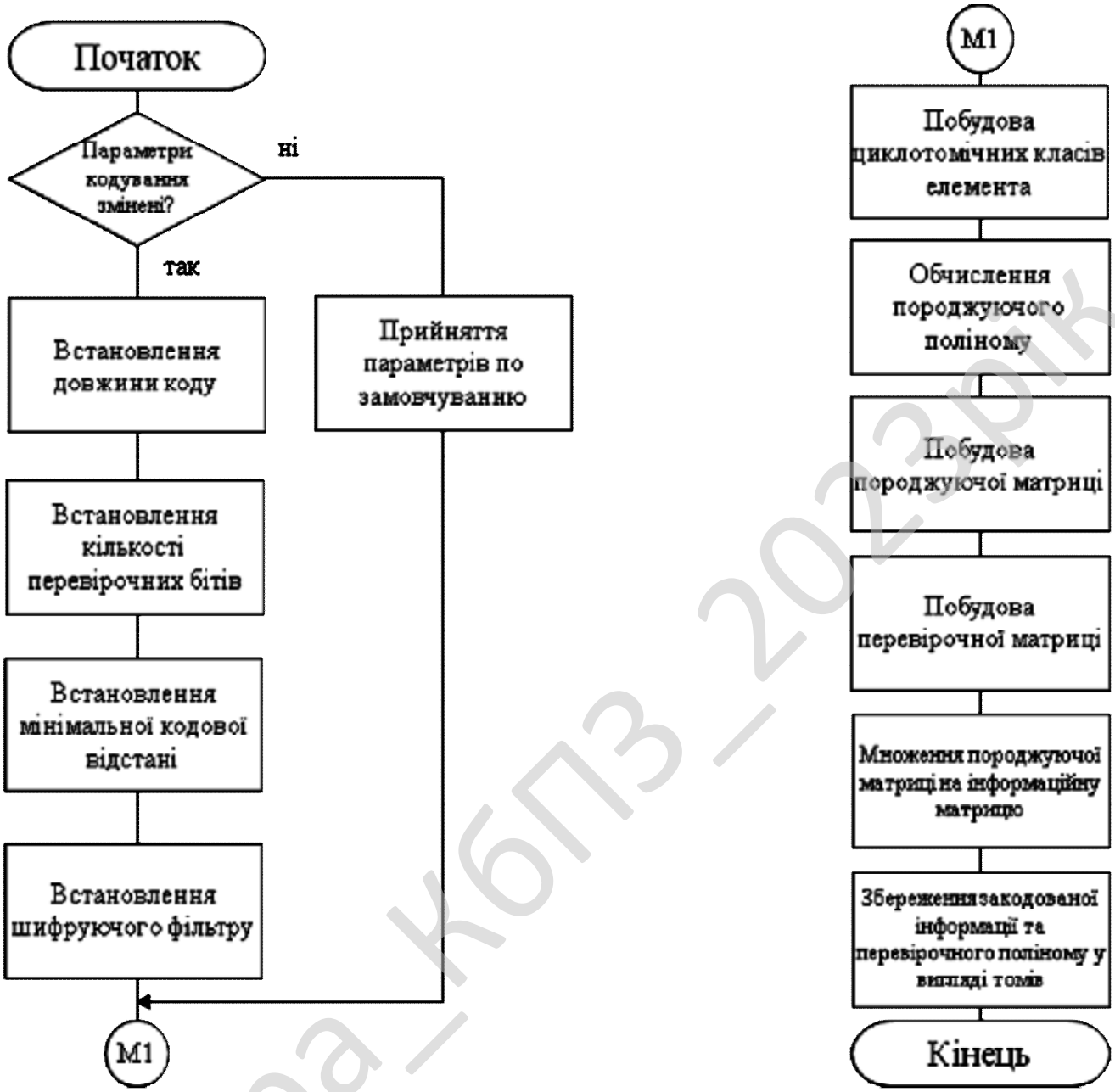

Рисунок 4.2 – Блок-схема роботи підпрограми кодування алгоритмом Ріда-Соломона

На рисунку 4.2 зображено блок-схему роботи підпрограми кодування алгоритмом Ріда-Соломона. З нього видно, що підпрограма працює наступним чином.

Спершу визначається, чи змінені параметри кодування.

Якщо вони змінені, тоді виконуються наступні дії:

– Встановлюється довжина коду.

– Встановлюється кількість перевірочних біт.

– Встановлюється мінімальна кодова відстань.

– Встановлюється шифруючий фільтр.

Якщо вони не змінені, тоді приймаються параметри кодування за замовчуванням.

Після цього починається процедура кодування інформації методом Ріда-Соломона. Вона виконується наступним чином:

– Будуються циклотомічні класи елемента.

– Обчислюється породжуючий поліном.

– Відбувається побудова породжуючої матриці.

– Відбувається побудова перевірочної матриці.

– Відбувається множення породжуючої матриці на інформаційну.

– Зберігається закодована інформація та перевірочний поліном, у вигляді томів.

На цьому підпрограма закінчує свою роботу.

На рисунку 4.3 зображено блок-схему роботи підпрограми декодування алгоритмом Ріда-Соломона. З нього видно, що підпрограма працює наступним чином.

Спершу відбувається читання томів.

Після цього прочитана інформація представляється у вигляді матриці.

Наступним кроком є множення інформаційної матриці на перевірочну матрицю.

Якщо усі коефіцієнти дорівнюють нулю, тоді виконуються наступні дії:

– Відновлюється та зберігається інформація.

– Виводиться повідомлення про те, що помилок не виявлено.

– Підпрограма завершує свою роботу.

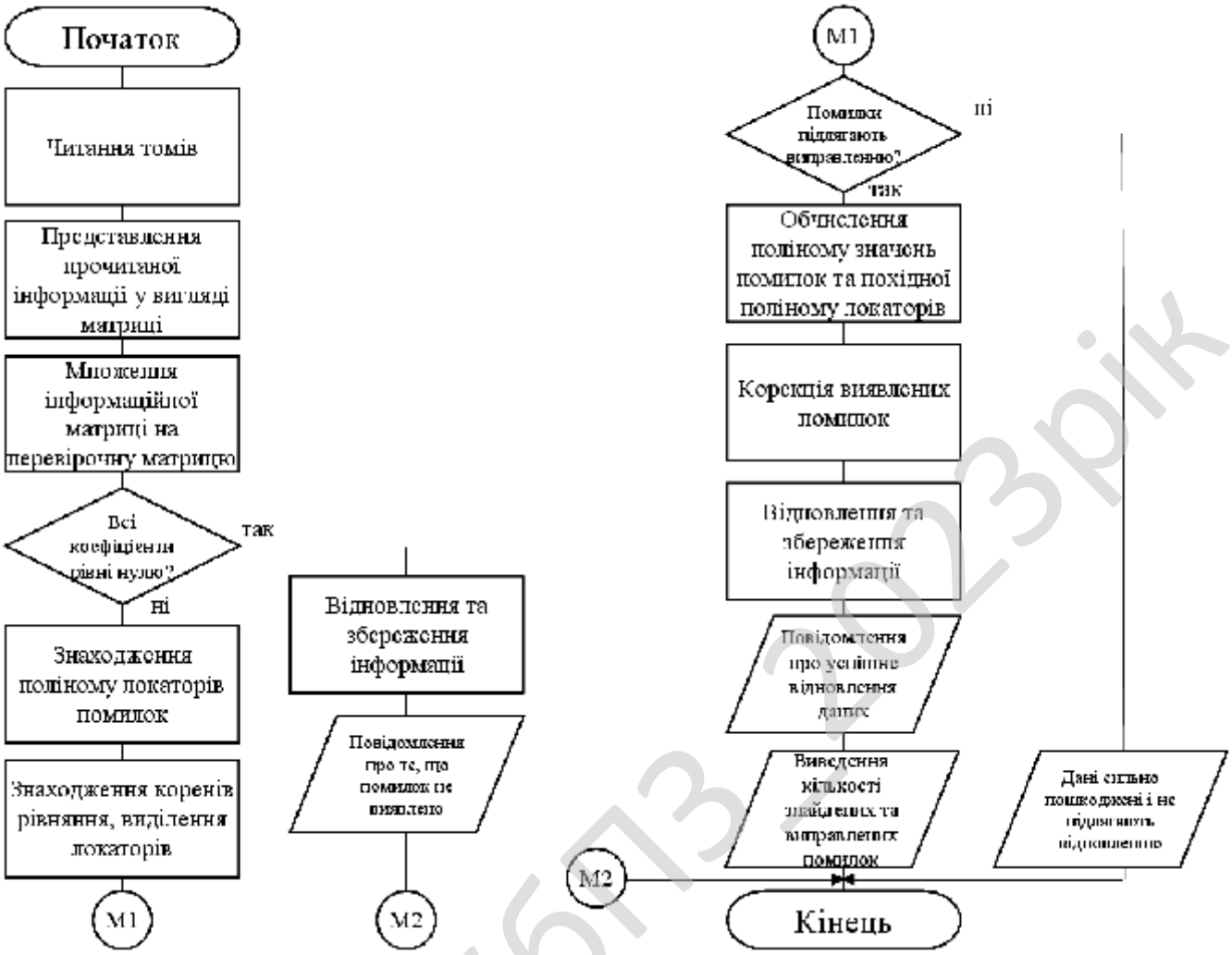

Рисунок 4.3 – Блок-схема роботи підпрограми декодування алгоритмом Ріда-Соломона

У іншому випадку, тобто, коли усі коефіцієнти не дорівнюють нулю, виконуються наступні дії:

– Знаходиться поліном локаторів помилок.

– Знаходяться корені рівнянь, виділяються локатори.

Якщо помилки підлягають виправленню, то виконуються наступні дії:

– Відбувається обчислення поліному значень помилок та похідної поліному локаторів.

– Відбувається корекція виявлених помилок.

– Відбувається відновлення та збереження інформації.

– Виводиться повідомлення про успішне відновлення даних.

– Виводиться повідомлення, у якому вказується кількість знайдених та виправлених помилок.

У іншому випадку, виводиться повідомлення, що дані сильно ушкоджені, й не підлягають відновленню, і на цьому підпрограма закінчує свою роботу.

Нижче наведемо ту частину коду, яка виконує вищеперераховані дії.

```
/*---------------------------------
*
* кодер Ріда-Соломона
* ======================
*
* кодуємі дані передаються через масив data[i], де i = 0...(k – 1),
* а згенеровані символи парності заносяться в масив b[0]...b[2 * t – 1].
* Вихідні й результуючі дані повинні бути представлені в поліноміальної
* формі (тобто у звичайній формі машинного подання даних).
* кодування виробляється з використанням сдвигового feedback-регістра,
* заповненого відповідними елементами масиву g[] з породженим
* поліномом усередині, процедура генерації якого вже обговорювалася в
* попередній главі.
* згенероване кодове слово описується наступною формулою:
* C(x) = data(x) * x ^ (n – k) + b(x), де ^ означає зведення в ступінь
*
-----------------------------------*/
encode_rs()
{
       int i, j;
       int feedback;
       // ініціалізація поля біт парності нулями
       for (i = 0; i < n – k; i++) b[i] = 0;
       // обробляємо всі символи
       // вихідних даних праворуч ліворуч
       for (i = k – 1; i >= 0; i-i--)
       {
// готовимо (data[i] + b[n – k – 1]) до множення на g[i]
// тобто складаємо черговий "захоплений" символ вихідних
// даних з молодшим символом біт парності (відповідного
// "регістру" b2 t-1, (рисунок 4.2) і переводимо його в
// індексну форму, зберігаючи результат у регістрі feedback;
```

```
// як ми вже говорили, сума двох індексів є добуток  поліномів
              feedback = index_of[data[i] ^ b[n - k - 1]];
 // є ще символи для обробки?
              if (feedback != -1)
              {
 // здійснюємо зрушення ланцюга bx-регістрів
                    for (j = n - k - 1; j > 0; j-j--)
  // якщо поточний коефіцієнт g - це дійсний
 // (тобто ненульовий коефіцієнт, те
  // множимо feedback на відповідний g-коефіцієнт
  // і складаємо його з наступних елементів ланцюжка
                          if (g[j] != -1) b[j] = b[j - 1] ^ alpha_to[(g[j] +
feedback) % n];
                    else
   // якщо поточний коефіцієнт g - це нульовий коефіцієнт,
   // виконуємо одне лише зрушення без множення, переміщаючи
   // символ з одного m-регістра в інший
                          b[j] = b[j - 1];
   // закільцьовуємо вихідний символ у крайній лівий b 0-регістр
                    b[0] = alpha_to[(g[0] + feedback) % n];
              }
              else
              {
  // розподіл завершений,
  // здійснюємо останнє зрушення регістра,
  // на виході регістра буде частка, що губиться,
  // а в самому регістрі - шуканий залишок
                    for (j = n - k - 1; j > 0; j-j--) b[j] = b[j - 1]; b[0] = 0;
              }
       }
}
```

Нижче наведемо вихідний текст підпрограми декодера Ріда-Соломона.

```
/*-----------------------------
* декодер Ріда-Соломона
* =====================
* Процедура декодування кодів Ріда-Соломона складається з декількох кроків:
* спочатку ми обчислюємо 2 t-символьний синдром шляхом постановки alpha**i в
* recd(x), де recd - отримане кодове слово, попередньо переведене
* в індексну форму. По факті обчислення recd(x) ми записуємо черговий
* символ синдрому в s[i], де і приймає значення від 1 до 2t, залишаючи
* s[0] рівним нулю.
```

```
* потім, використовуючи ітеративний алгоритм Берлекэмпа (Berlekamp), ми
* знаходимо поліном локатора помилки – elp[i]. Якщо ступінь elp перевищує
* собою величину t, ми неспроможні скорегувати всі помилки й обмежуємося
* виводом повідомлення про непереборну помилку, після чого робимо аварійний
* вихід з декодера. Якщо ж ступінь elp не перевищує t, ми підставляємо
* alpha**i, де i = 1...n в elp для обчислення корінь полінома. Обіг
* знайдений корінь дає нам позиції перекручених символів. Якщо кількість
* певних позицій перекручених символів менше ступеня elp, перекручуванню
* піддалося більш ніж t символів і ми не можемо відновити їх.
* У всіх інших випадках відновлення оригінального вмісту
* перекручених символів цілком можливо.
* У випадку, коли кількість помилок свідомо велико, для їхнього виправлення
* декодуємі символи проходять крізь декодер без яких або змін
-------------------------------------*/
decode_rs()
{
       int i, j, u, q;
       int s[n – k + 1]; // поліном синдрому помилки
       int elp[n – k + 2][n – k]; // поліном локатора помилки лямда
       int d[n – k + 2];
       int l[n – k + 2];
       int u_lu[n – k + 2],
       int count = 0, syn_error = 0, root[t], loc[t], z[t + 1], err[n], reg[t +
1];
       // переводимо отримане кодове слово в індексну форму
       // для спрощення обчислень
       for (i = 0; i < n; i++) recd[i] = index_of[recd[i]];
       // обчислюємо синдром
       //---------------------------------
       for (i = 1; i <= n – k; i++)
       {
            s[i] = 0;        // ініціалізація s-регістра
                             // на його вхід за замовчуванням надходить нуль
// виконуємо s[i] += recd[j] * ij
// тобто беремо черговий символ декодуємих даних,
// множимо його на порядковий номер даного символу,
// помножений на номер чергового оберту й складаємо
// отриманий результат із умістом s-регістра;
// по факті вичерпання всіх декодуємих символ ми
// повторюємо весь цикл обчислень знову – по одному
// разу для кожного символу парності
             for (j = 0; j < n; j++) if (recd[j] != -1) s[i] ^= alpha_to[(recd[j]
```

```
+ i * j) % n];
                if (s[i] != 0) syn_error = 1; // якщо синдром не дорівнює нулю,
зводимо прапор помилки
// перетворимо синдром з поліноміальної форми в індексну
                s[i] = index_of[s[i]];
        }
        // корекція помилок
        //------------------------
        if (syn_error)                // якщо є помилки, намагаємося їх скорегувати
        {
 // обчислення полінома локатора ламбда
//---------------------------
// обчислюємо поліном локатора помилки через ітеративний алгоритм
// Берлекемпа. Випливаючи термінології Lin and Costello (див. "Error
// Control Coding: Fundamentals and Applications" Prentice Hall 1983
// ISBN 013283796) d[u] являє собою m ("мю"), що виражає
// розбіжність (discrepancy), де u = m + 1 і m є номер кроку
// з діапазону від -1 до 2t. У Блейхута та ж сама величина
// позначається D(x) ("дельта") і називається нев'язання.
// l[u] являє собою ступінь elp для даного кроку ітерації,
// u_l[u] являє собою різницю між номером кроку й ступенем elp
                // ініціалізація елементи таблиці
                d[0] = 0; // індексна форма
                d[1] = s[1]; // індексна форма
                elp[0][0] = 0; // індексна форма
                elp[1][0] = 1; // поліноміальна форма
                for (i = 1; i < n - k; i++)
                {
                        elp[0][i] = -1; // індексна форма
                        elp[1][i] = 0; // поліноміальна форма
                }
                l[0] = 0; l[1] = 0; u_lu[0] = -1; u_lu[1] = 0; u = 0;
                do
                {
                        u++;
                        if (d[u] == -1)
                        {
                                l[u + 1] = l[u];
                                for (i = 0; i <= l[u]; i++)
                                {
                                        elp[u + 1][i] = elp[u][i];
                                        elp[u][i] = index_of[elp[u][i]];
```

```
                        }
                    }
                        else
                    {
                        // пошук слів з найбільшим u_lu[q], таких що d[q] != 0
                        q = u - 1;
                        while ((d[q] == -1) && (q > 0)) q-q--;
                        // знайдений перший ненульовий d[q]
                        if (q > 0)
                        {
                            j = q;
                            do
                        {
                            j-j-- ;
                            if ((d[j] != -1) && (u_lu[q] < u_lu[j]))
                            q = j;
                        } while (j > 0);
                    };
 // як тільки ми знайдемо q, такий що d[u] !=0
// і u_lu[q] є максимум
// запишемо ступінь нового elp полінома
                    if (l[u] > l[q] + u - q) l[u + 1] = l[u]; else l[u + 1] =
l[q] + u - q;
// формуємо новий elp(x)
                    for (i = 0; i < n - k; i++) elp[u + 1][i] = 0;
                    for (i = 0; i <= l[q]; i++)
                        if (elp[q][i] != -1)
                            elp[u + 1][i + u - q] = alpha_to[(d[u] + n -
 d[q] + elp[q][i]) % n];
                    for (i = 0; i <= l[u]; i++)
                    {
                        elp[u + 1][i] ^= elp[u][i];
                        // перетворимо старий elp
                        // в індексну форму
                        elp[u][i] = index_of[elp[u][i]];
                    }
                }
                u_lu[u + 1] = u - l[u + 1];
 // формуємо (u + 1)'ю нев'язання
//---------------------------
                if (u < n - k)          // на останній ітерації розбіжність
                {                       // не було виявлено
```

```
                    if (s[u + 1] != -1) d[u + 1] = alpha_to[s[u + 1]]; else d[u +
1] = 0;
                    for (i = 1; i <= l[u + 1]; i++)
                           if ((s[u + 1 - i] != -1) && (elp[u + 1][i] != 0))
                           d[u + 1] ^= alpha_to[(s[u + 1 - i] + index_of[elp[u +
1][i]]) % n];
                    // переводимо d[u + 1] в індексну форму
                    d[u + 1] = index_of[d[u + 1]];
             }
      } while ((u < n - k) && (l[u + 1] <= t));
      // розрахунок локатора завершений
      //-------------------------
      u++ ;
      if (l[u] <= t)
      {
             // корекція помилок можлива
             // переводимо elp в індексну форму
             for (i = 0; i <= l[u]; i++) elp[u][i] = index_of[elp[u][i]];
// знаходження корінь полінома локатора помилки
//---------------------------
             for (i = 1; i <= l[u]; i++) reg[i] = elp[u][i]; count = 0;
             for (i = 1; i <= n; i++)
             {
                    q = 1 ;
                    for (j = 1; j <= l[u]; j++)
                    if (reg[j] != -1)
                    {
                           reg[j] = (reg[j] + j) % n;
                           q ^= alpha_to[reg[j]];
                    }
                    if (!q)
                    { // записуємо корінь і індекс позиції помилки
                           root[count] = i;
                           loc[count] = n - i;
                           count++;
                    }
             }
             if (count == l[u])
             { // немає корінь - ступінь elp < t помилок
                    // формуємо поліном z(x)
                    for (i = 1; i <= l[u]; i++) // Z[0] завжди дорівнює 1
                    {
```

```
                        if ((s[i] != -1) && (elp[u][i] != -1))
                                z[i] = alpha_to[s[i]] ^ alpha_to[elp[u][i]];
                        else
                                if ((s[i]! =- 1) && (elp[u][i] == -1))
                                        z[i] = alpha_to[s[i]];
                                else
                                        if ((s[i]==-1) && (elp[u][i]!=-1))
                                                z[i] = alpha_to[elp[u][i]];
                                        else
                                                        z[i] = 0 ;
                for (j = 1; j < i; j++)
                        if ((s[j] != -1) && (elp[u][i - j] != -1))
                                z[i] ^= alpha_to[(elp[u][i - j] + s[j]) % n];
                // переводимо z[i] в індексну форму
                z[i] = index_of[z[i]];
        }
// обчислення значення помилок у позиціях loc[i]
//---------------------------
        for (i = 0; i < n; i++)
        {
                err[i] = 0;
                // переводимо recd[] у поліноміальну форму
                if (recd[i] != -1) recd[i] = alpha_to[recd[i]]; else recd[i]
= 0;
        }
        // спочатку обчислюємо чисельник помилки
        for (i = 0; i < l[u]; i++)
        {
                err[loc[i]] = 1;
                for (j = 1; j <= l[u]; j++)
                        if (z[j] != -1)
                                err[loc[i]] ^= alpha_to[(z[j] + j * root[i]) %
n];
                if (err[loc[i]] != 0)
                {
                        err[loc[i]] = index_of[err[loc[i]]];
                        q = 0 ; // формуємо знаменник коефіцієнта помилки
                        for (j = 0; j < l[u]; j++)
                                if (j != i)
                                 q += index_of[1 ^ alpha_to[(loc[j] + root[i])
% n]];
                        q = q % n; err[loc[i]] = alpha_to[(err[loc[i]] - q +
```

```
n) % n];
                                // recd[i] повинен бути в поліноміальній формі
                                recd[loc[i]] ^= err[loc[i]];
                        }
                }
        }
                else // немає корінь,
                        // рішення системи рівнянь неможливо, тому що ступінь elp >=
t
        {
                // переводимо recd[] у поліноміальну форму
                for (i = 0; i < n; i++)
                        if (recd[i] != -1) recd[i] = alpha_to[recd[i]];
                else
                        recd[i] = 0; // виводимо інформаційне слово як є
        }
                else // ступінь elp > t, рішення неможливо
        {
        // переводимо recd[] у поліноміальну форму
        for (i = 0; i < n; i++)
                if (recd[i] != -1)
                        recd[i] = alpha_to[recd[i]];
                else
                        recd[i] = 0 ; // виводимо інформаційне слово як є
        }
                else                        // помилок не виявлено
        for (i = 0; i < n; i++) if (recd[i] != -1) recd[i] = alpha_to[recd[i]];
else recd[i] = 0;
}
```

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою алгоритму захисту інформації RSA. Спочатку необхідно обчислити пару ключів (секретний ключ і відкритий ключ). Для цього відправник електронних документів обчислює два більших простих числа P и Q, потім знаходить їхній добуток $N = P*Q$ і значення функції $\varphi(N) = (P-1)(Q-1)$. Далі відправник обчислює число E з умов $E < \varphi(N)$, НЗД $(E, \varphi(N)) = 1$ і число D з умов $D < N$, $E*D \equiv 1 (\mathrm{mod}\ \varphi(N))$.

Пари чисел (E, N) є відкритим ключем. Цю пару чисел автор передає партнерам по переписці для перевірки його цифрових підписів. Число D зберігається автором як секретний ключ для підписування.

Допустимо, що відправник хоче підписати повідомлення M перед його відправленням. Спочатку повідомлення M (блок інформації, файл, таблиця) стискають за допомогою геш-функції h(-) у ціле число m: $m = h(M)$.

Потім обчислюють цифровий підпис S під електронним документом M, використовуючи геш-значення m і секретний ключ D: $S = m \pmod N$.

Пари (M, S) передається партнерові-одержувачеві як електронний документ M, підписаний цифровим підписом S, причому підпис S сформований власником секретного ключа D.

Після прийому пари (M, S) одержувач обчислює геш-значення повідомлення M двома різними способами. Насамперед, він відновлює геш-значення m', застосовуючи криптографічне перетворення підпису S з використанням відкритого ключа E: $m' = S^E \pmod N$.

Крім того, він знаходить результат гешування прийнятого повідомлення M с допомогою такої ж геш-функції h(-): $m = h(M)$.

Якщо дотримується рівність обчислених значень, тобто $S^E(\text{mod}) = h(M)$, то одержувач визнає пару (M, S) справжньою. Доведено, що тільки власник секретного ключа D може сформувати цифровий підпис S по документі M, а визначити секретне число D по відкритому числу E не легше, ніж розкласти модуль N на множники. Крім того, можна строго математично довести, що результат перевірки цифрового підпису S буде позитивним тільки в тому випадку, якщо при обчисленні S був використаний секретний ключ D, що відповідає відкритому ключу E. Тому відкритий ключ E іноді називають "ідентифікатором" того, хто підписав.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено основне вікно програми. Воно складається з наступних блоків:

– Блок меню.

– Блок кнопок швидкого доступу до елементів програми.

– Блок доступу до даних які потрібно зберігати, та до даних, які вже в томах.

Блок меню складається з наступних елементів:

– Файл.

– Дата-центр.

– Збереження.

– Відновлення.

– Тестування.

– Параметри.

– Довідка.

Блок кнопок швидкого доступу до елементів програми складається з наступних елементів:

– Зберегти дані.

– Відновити дані.

– Пошук помилок.

– Виправлення помилок.

– Введення надлишковості.

– Вибір кількості дисків.

На рисунку 5.2 зображено вікно довідки, з якого стає зрозумілим, хто розробник програми, керівник проекту, тема проекту, та місце виконання проекту.

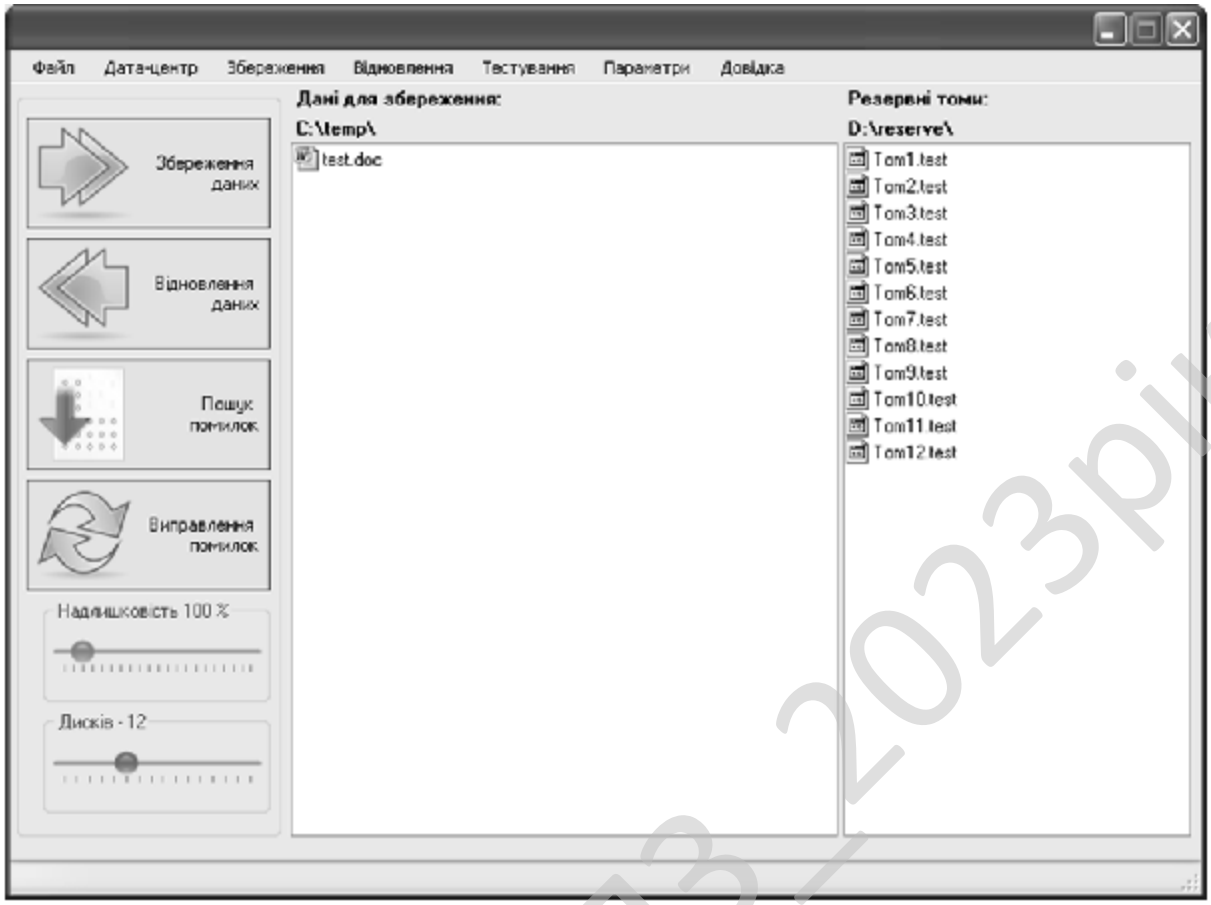

Рисунок 5.1 – Основне вікно програми

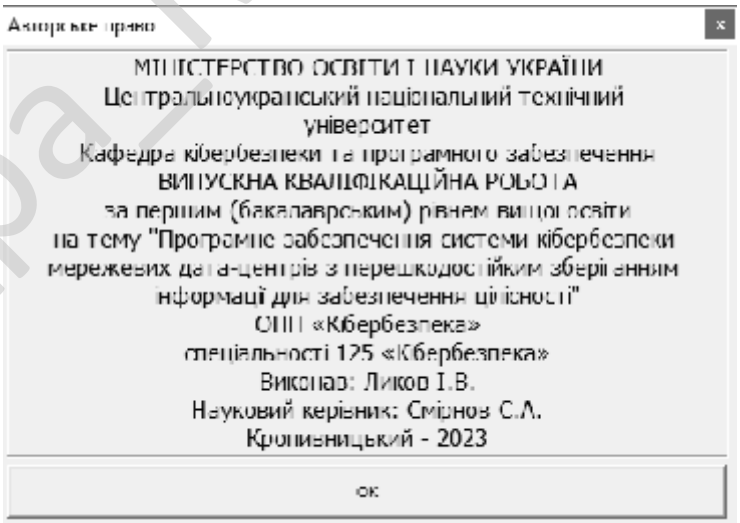

Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

– Досліджена система мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

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 створені 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 кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності. Це 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 кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм RSA.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022, pp. 1-12. **(Scopus).**

2. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz 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. (**Scopus**).

3. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. **Springer**, Singapore. pp. 21-34. **(Scopus).**

4. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. **Springer**, Cham. 2022, pp. 2463-2477. **(Scopus).**

5. 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 **(Scopus).**

6. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T),* 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus).**

7. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

8. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.* **(Scopus)**.

9. 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. **(Scopus)**.

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

11. Smirnov O.A., 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. **(Scopus)**.

12. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131. **(Scopus)**.

13. 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. **(Scopus)**.

14. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. **Springer**, Cham. 2021, pp 66-84. **(Scopus)**.

15. 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. **(Scopus)**.

16. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136. **(Scopus)**.

17. 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. **(Scopus)**.

18. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43. **(Scopus)**.

19. 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. **(Scopus)**.

20. 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., **(Scopus)**.

21. 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. (**Scopus)**.

22. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019. **(Scopus)**.

23. 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. **(Scopus)**.

24. 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. **(Scopus)**.

25. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884. **(Scopus)**.

26. Smirnov, O., Kuznetsov, A., Prokopovych-Tkachenko, D. «Hiding Data in Images Using a Pseudo-Random Sequence». *ISCI'2020: Information Security in Critical Infrastructures.* ***Collective monograph***. Edited by Ivan D. Gorbenko, Victor A. Krasnobayev and Alexandr A. Kuznetsov. ASC Academic Publishing, USA, 2020. pp. 46-59. – ISBN: 978-1-7362833-0-1 (Hardback), ISBN: 978-1-7362833-1-8 (Ebook).

27. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: 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. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

28. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: 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).

29. 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.

30. Смірнов О.А., Дрєєва Г.М., «Метод генерування фрактального трафіку за допомогою моделі генератора на графі» у Інформаційна безпека та інформаційні технології: монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139.

31. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования WEB-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

32. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

33. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98. 2022.

34. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

35. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

36. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного

процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

37. Смирнов А., Кузнецов А., Кузнецова Т. «Шумоподобные дискретные сигналы для асинхронных систем кодового разделения радиоканалов». *Радиотехника*, № 2(205), 175–183. 2021.

38. 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.

39. Смірнов, О.А., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю.Усік П.С., «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». *Проблеми телекомунікацій*. № 1(26). С. 83-96. 2020.

40. Смирнов А.А, Кузнецов А.А., Киян А.С., Кузнецова Е.А. «Сокрытие данных на основе адресации шумоподобных сигналов». *Всеукраїнський міжвідомчий науково-технічний збірник "Радіотехніка"* – Харьків: ХНУРЕ. – 2020. – Вип. 203. – С. 38-49.

41. Смирнов А.А., Дудан А.В., Смирнова Т.В. «Формализация структуры технологического процесса электродугового напыления». *Сборник научных трудов «Актуальные вопросы машиноведения».* Объединенный институт машиностроения Национальной Академии Наук Беларуси. №9. С. 308-312, 2020.

42. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

43. 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.

44. А.А. Смирнов, Т.В. Смирнова, А.Н. Дреев, А.В. Дудан. «Оптимизация технологического процесса восстановления и упрочнения поверхностей с заданными характеристиками в виде облачного сервиса». Вестник Полоцкого государственного университета. Серия В, Промышленность. Прикладные науки. Республика Беларусь - 2020. - № 3. - С. 50-61.

45. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

46. Смірнов О.А., Смірнова Т.В., Солових Є.К., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

47. Смірнов О.А., Смірнова Т.В.,. Дрєєв О.М, «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», Системи управління, навігації та зв'язку, № 2 (54). с. 149-154, 2019.

48. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

49. Смирнов А.А., Лысенко И.А., Информационная технология проектирования тестовых наборов на основе требований к программному обеспечению, Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

50. Смірнов О.А., Мелешко Є.В., Хох В.Д., Дослідження методів аудиту систем управління інформаційною безпекою, Системи управління, навігації та зв'язку. – Випуск 1 (41). – Полтава: ПолтНТУ. – 2017. – С. 38-42.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 13-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Visual C#.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 80 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки мережевих дата-центрів з перешкодостійким зберіганням інформації для забезпечення цілісності

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 56

Літера: РП

Кропивницький – 2023 року

Файл ProcessForm.cs - вікно відображення процесів запису/читання та перевірки цілісності даних у дата-центрі

```
namespace Data_Center
{
    partial class ProcessForm
    {
        /// <summary>
                /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true якщо керуючі ресурси повинні бути
розташовані, у іншому випадку  false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Необхідний метод для підтримки розробника - не модифікується
        /// зміст цього метода використовується редактором коду.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(ProcessForm));
            this.processPriorityGroupBox = new System.Windows.Forms.GroupBox();
            this.processPriorityComboBox = new System.Windows.Forms.ComboBox();
            this.processGroupBox = new System.Windows.Forms.GroupBox();
            this.processProgressBar = new System.Windows.Forms.ProgressBar();
            this.fileAnalyzeStatGroupBox = new System.Windows.Forms.GroupBox();
            this.percOfAltEccLabel = new System.Windows.Forms.Label();
            this.percOfDamageLabel = new System.Windows.Forms.Label();
            this.percOfAltEccLabel_ = new System.Windows.Forms.Label();
            this.percOfDamageLabel_ = new System.Windows.Forms.Label();
            this.logGroupBox = new System.Windows.Forms.GroupBox();
            this.logListBox = new System.Windows.Forms.ListBox();
            this.countGroupBox = new System.Windows.Forms.GroupBox();
            this.errorCountLabel = new System.Windows.Forms.Label();
            this.okCountLabel = new System.Windows.Forms.Label();
            this.errorPictureBox = new System.Windows.Forms.PictureBox();
            this.okPictureBox = new System.Windows.Forms.PictureBox();
            this.errorCountLabel_ = new System.Windows.Forms.Label();
            this.okCountLabel_ = new System.Windows.Forms.Label();
            this.toolTip = new System.Windows.Forms.ToolTip(this.components);
            this.stopButtonXP = new PinkieControls.ButtonXP();
            this.pauseButtonXP = new PinkieControls.ButtonXP();
            this.closingTimer = new System.Windows.Forms.Timer(this.components);
            this.processTimer = new System.Windows.Forms.Timer(this.components);
            this.processPriorityGroupBox.SuspendLayout();
            this.processGroupBox.SuspendLayout();
            this.fileAnalyzeStatGroupBox.SuspendLayout();
            this.logGroupBox.SuspendLayout();
            this.countGroupBox.SuspendLayout();
```

```
((System.ComponentModel.ISupportInitialize)(this.errorPictureBox)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.okPictureBox)).BeginInit();
            this.SuspendLayout();
            //
            // processPriorityGroupBox
            //

this.processPriorityGroupBox.Controls.Add(this.processPriorityComboBox);
            this.processPriorityGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.processPriorityGroupBox.Location = new
System.Drawing.Point(617, 216);
            this.processPriorityGroupBox.Name = "processPriorityGroupBox";
            this.processPriorityGroupBox.Size = new System.Drawing.Size(135,
64);
            this.processPriorityGroupBox.TabIndex = 0;
            this.processPriorityGroupBox.TabStop = false;
            this.processPriorityGroupBox.Text = "Пріоритет процесу";
            //
            // processPriorityComboBox
            //
            this.processPriorityComboBox.BackColor =
System.Drawing.SystemColors.Control;
            this.processPriorityComboBox.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.processPriorityComboBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.processPriorityComboBox.FormattingEnabled = true;
            this.processPriorityComboBox.Items.AddRange(new object[] {
            "За замовчуванням",
            "Знижений",
            "Нормальний",
            "Підвищений",
            "Найвищий"});
            this.processPriorityComboBox.Location = new System.Drawing.Point(9,
33);
            this.processPriorityComboBox.Name = "processPriorityComboBox";
            this.processPriorityComboBox.Size = new System.Drawing.Size(117,
21);
            this.processPriorityComboBox.TabIndex = 0;
            this.processPriorityComboBox.TabStop = false;
            this.toolTip.SetToolTip(this.processPriorityComboBox, "Список
можливих значень пріоритету процесу обробки");
            this.processPriorityComboBox.SelectedIndexChanged += new
System.EventHandler(this.processPriorityComboBox_SelectedIndexChanged);
            //
            // processGroupBox
            //
            this.processGroupBox.Controls.Add(this.processProgressBar);
            this.processGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.processGroupBox.Location = new System.Drawing.Point(12, 9);
            this.processGroupBox.Name = "processGroupBox";
            this.processGroupBox.Size = new System.Drawing.Size(871, 65);
            this.processGroupBox.TabIndex = 0;
            this.processGroupBox.TabStop = false;
            this.processGroupBox.Text = "Обробка";
            //
            // processProgressBar
            //
            this.processProgressBar.Location = new System.Drawing.Point(14, 30);
            this.processProgressBar.Name = "processProgressBar";
            this.processProgressBar.Size = new System.Drawing.Size(844, 20);
            this.processProgressBar.Style =
System.Windows.Forms.ProgressBarStyle.Continuous;
            this.processProgressBar.TabIndex = 0;
            //
```

```
            // fileAnalyzeStatGroupBox
            //
            this.fileAnalyzeStatGroupBox.Controls.Add(this.percOfAltEccLabel);
            this.fileAnalyzeStatGroupBox.Controls.Add(this.percOfDamageLabel);
            this.fileAnalyzeStatGroupBox.Controls.Add(this.percOfAltEccLabel_);
            this.fileAnalyzeStatGroupBox.Controls.Add(this.percOfDamageLabel_);
            this.fileAnalyzeStatGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.fileAnalyzeStatGroupBox.Location = new System.Drawing.Point(12,
216);
            this.fileAnalyzeStatGroupBox.Name = "fileAnalyzeStatGroupBox";
            this.fileAnalyzeStatGroupBox.Size = new System.Drawing.Size(459,
64);
            this.fileAnalyzeStatGroupBox.TabIndex = 0;
            this.fileAnalyzeStatGroupBox.TabStop = false;
            this.fileAnalyzeStatGroupBox.Text = "Результат аналізу цілісності
даних";
            //
            // percOfAltEccLabel
            //
            this.percOfAltEccLabel.AutoSize = true;
            this.percOfAltEccLabel.Location = new System.Drawing.Point(195, 41);
            this.percOfAltEccLabel.Name = "percOfAltEccLabel";
            this.percOfAltEccLabel.Size = new System.Drawing.Size(10, 13);
            this.percOfAltEccLabel.TabIndex = 0;
            this.percOfAltEccLabel.Text = "-";
            //
            // percOfDamageLabel
            //
            this.percOfDamageLabel.AutoSize = true;
            this.percOfDamageLabel.Location = new System.Drawing.Point(195, 20);
            this.percOfDamageLabel.Name = "percOfDamageLabel";
            this.percOfDamageLabel.Size = new System.Drawing.Size(10, 13);
            this.percOfDamageLabel.TabIndex = 0;
            this.percOfDamageLabel.Text = "-";
            //
            // percOfAltEccLabel_
            //
            this.percOfAltEccLabel_.AutoSize = true;
            this.percOfAltEccLabel_.Location = new System.Drawing.Point(7, 41);
            this.percOfAltEccLabel_.Name = "percOfAltEccLabel_";
            this.percOfAltEccLabel_.Size = new System.Drawing.Size(163, 13);
            this.percOfAltEccLabel_.TabIndex = 0;
            this.percOfAltEccLabel_.Text = "Резерв перевірочних даних для
відновлення:";
            //
            // percOfDamageLabel_
            //
            this.percOfDamageLabel_.AutoSize = true;
            this.percOfDamageLabel_.Location = new System.Drawing.Point(7, 20);
            this.percOfDamageLabel_.Name = "percOfDamageLabel_";
            this.percOfDamageLabel_.Size = new System.Drawing.Size(148, 13);
            this.percOfDamageLabel_.TabIndex = 0;
            this.percOfDamageLabel_.Text = "Всього пошкоджених секторів:";
            //
            // logGroupBox
            //
            this.logGroupBox.Controls.Add(this.logListBox);
            this.logGroupBox.Location = new System.Drawing.Point(12, 80);
            this.logGroupBox.Name = "logGroupBox";
            this.logGroupBox.Size = new System.Drawing.Size(871, 130);
            this.logGroupBox.TabIndex = 0;
            this.logGroupBox.TabStop = false;
            this.logGroupBox.Text = "Лог процесу";
            //
            // logListBox
            //
            this.logListBox.BackColor = System.Drawing.SystemColors.Control;
            this.logListBox.BorderStyle = System.Windows.Forms.BorderStyle.None;
```

```
            this.logListBox.FormattingEnabled = true;
            this.logListBox.HorizontalScrollbar = true;
            this.logListBox.Location = new System.Drawing.Point(7, 23);
            this.logListBox.Name = "logListBox";
            this.logListBox.SelectionMode =
System.Windows.Forms.SelectionMode.None;
            this.logListBox.Size = new System.Drawing.Size(851, 91);
            this.logListBox.TabIndex = 0;
            this.logListBox.TabStop = false;
            this.logListBox.UseTabStops = false;
            this.logListBox.SelectedIndexChanged += new
System.EventHandler(this.logListBox_SelectedIndexChanged);
            //
            // countGroupBox
            //
            this.countGroupBox.Controls.Add(this.errorCountLabel);
            this.countGroupBox.Controls.Add(this.okCountLabel);
            this.countGroupBox.Controls.Add(this.errorPictureBox);
            this.countGroupBox.Controls.Add(this.okPictureBox);
            this.countGroupBox.Controls.Add(this.errorCountLabel_);
            this.countGroupBox.Controls.Add(this.okCountLabel_);
            this.countGroupBox.Location = new System.Drawing.Point(482, 216);
            this.countGroupBox.Name = "countGroupBox";
            this.countGroupBox.Size = new System.Drawing.Size(124, 64);
            this.countGroupBox.TabIndex = 0;
            this.countGroupBox.TabStop = false;
            this.countGroupBox.Text = "Лічильник процесу";
            //
            // errorCountLabel
            //
            this.errorCountLabel.AutoSize = true;
            this.errorCountLabel.Location = new System.Drawing.Point(63, 41);
            this.errorCountLabel.Name = "errorCountLabel";
            this.errorCountLabel.Size = new System.Drawing.Size(13, 13);
            this.errorCountLabel.TabIndex = 0;
            this.errorCountLabel.Text = "0";
            this.toolTip.SetToolTip(this.errorCountLabel, "Лічильник некоректно
оброблених файлів");
            //
            // okCountLabel
            //
            this.okCountLabel.AutoSize = true;
            this.okCountLabel.Location = new System.Drawing.Point(63, 20);
            this.okCountLabel.Name = "okCountLabel";
            this.okCountLabel.Size = new System.Drawing.Size(13, 13);
            this.okCountLabel.TabIndex = 0;
            this.okCountLabel.Text = "0";
            this.toolTip.SetToolTip(this.okCountLabel, "Лічильник коректно
оброблених файлів");
            //
            // errorPictureBox
            //
            this.errorPictureBox.Image =
((System.Drawing.Image)(resources.GetObject("errorPictureBox.Image")));
            this.errorPictureBox.Location = new System.Drawing.Point(10, 40);
            this.errorPictureBox.Name = "errorPictureBox";
            this.errorPictureBox.Size = new System.Drawing.Size(21, 15);
            this.errorPictureBox.TabIndex = 2;
            this.errorPictureBox.TabStop = false;
            //
            // okPictureBox
            //
            this.okPictureBox.Image =
((System.Drawing.Image)(resources.GetObject("okPictureBox.Image")));
            this.okPictureBox.Location = new System.Drawing.Point(10, 19);
            this.okPictureBox.Name = "okPictureBox";
            this.okPictureBox.Size = new System.Drawing.Size(21, 15);
            this.okPictureBox.TabIndex = 1;
            this.okPictureBox.TabStop = false;
```

```
            //
            // errorCountLabel_
            //
            this.errorCountLabel_.AutoSize = true;
            this.errorCountLabel_.Location = new System.Drawing.Point(28, 41);
            this.errorCountLabel_.Name = "errorCountLabel_";
            this.errorCountLabel_.Size = new System.Drawing.Size(35, 13);
            this.errorCountLabel_.TabIndex = 0;
            this.errorCountLabel_.Text = "Error :";
            this.toolTip.SetToolTip(this.errorCountLabel_, "Лічильник некоректно
оброблених файлів");
            //
            // okCountLabel_
            //
            this.okCountLabel_.AutoSize = true;
            this.okCountLabel_.Location = new System.Drawing.Point(28, 20);
            this.okCountLabel_.Name = "okCountLabel_";
            this.okCountLabel_.Size = new System.Drawing.Size(28, 13);
            this.okCountLabel_.TabIndex = 0;
            this.okCountLabel_.Text = "OK :";
            this.toolTip.SetToolTip(this.okCountLabel_, "Лічильник коректно
оброблених файлів");
            //
            // toolTip
            //
            this.toolTip.AutomaticDelay = 2000;
            this.toolTip.AutoPopDelay = 20000;
            this.toolTip.InitialDelay = 2000;
            this.toolTip.ReshowDelay = 1000;
            //
            // stopButtonXP
            //
            this.stopButtonXP.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(236)))),
((int)(((byte)(233)))), ((int)(((byte)(216)))));
            this.stopButtonXP.DefaultScheme = true;
            this.stopButtonXP.DialogResult =
System.Windows.Forms.DialogResult.None;
            this.stopButtonXP.Hint = "";
            this.stopButtonXP.Location = new System.Drawing.Point(762, 257);
            this.stopButtonXP.Name = "stopButtonXP";
            this.stopButtonXP.Scheme = PinkieControls.ButtonXP.Schemes.Blue;
            this.stopButtonXP.Size = new System.Drawing.Size(121, 23);
            this.stopButtonXP.TabIndex = 2;
            this.stopButtonXP.Text = "Перервати обробку";
            this.toolTip.SetToolTip(this.stopButtonXP, "Припинення обробки
файлів із закриттям даного вікна");
            this.stopButtonXP.Click += new
System.EventHandler(this.stopButtonXP_Click);
            //
            // pauseButtonXP
            //
            this.pauseButtonXP.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(236)))),
((int)(((byte)(233)))), ((int)(((byte)(216)))));
            this.pauseButtonXP.DefaultScheme = true;
            this.pauseButtonXP.DialogResult =
System.Windows.Forms.DialogResult.None;
            this.pauseButtonXP.Hint = "";
            this.pauseButtonXP.Location = new System.Drawing.Point(762, 220);
            this.pauseButtonXP.Name = "pauseButtonXP";
            this.pauseButtonXP.Scheme = PinkieControls.ButtonXP.Schemes.Blue;
            this.pauseButtonXP.Size = new System.Drawing.Size(121, 23);
            this.pauseButtonXP.TabIndex = 1;
            this.pauseButtonXP.Text = "Пауза";
            this.toolTip.SetToolTip(this.pauseButtonXP, "Постановка/зняття
процесу обробки з паузи");
            this.pauseButtonXP.Click += new
System.EventHandler(this.pauseButtonXP_Click);
```

```
            //
            // closingTimer
            //
            this.closingTimer.Tick += new
System.EventHandler(this.closingTimer_Tick);
            //
            // processTimer
            //
            this.processTimer.Interval = 500;
            this.processTimer.Tick += new
System.EventHandler(this.processTimer_Tick);
            //
            // ProcessForm
            //
            this.AutoScaleDimensions = new System.Drawing.Size(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(894, 292);
            this.ControlBox = false;
            this.Controls.Add(this.stopButtonXP);
            this.Controls.Add(this.pauseButtonXP);
            this.Controls.Add(this.countGroupBox);
            this.Controls.Add(this.processPriorityGroupBox);
            this.Controls.Add(this.logGroupBox);
            this.Controls.Add(this.fileAnalyzeStatGroupBox);
            this.Controls.Add(this.processGroupBox);
            this.DoubleBuffered = true;
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "ProcessForm";
            this.ShowInTaskbar = false;
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Обробка файлу";
            this.Load += new System.EventHandler(this.ProcessForm_Load);
            this.processPriorityGroupBox.ResumeLayout(false);
            this.processGroupBox.ResumeLayout(false);
            this.fileAnalyzeStatGroupBox.ResumeLayout(false);
            this.fileAnalyzeStatGroupBox.PerformLayout();
            this.logGroupBox.ResumeLayout(false);
            this.countGroupBox.ResumeLayout(false);
            this.countGroupBox.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.errorPictureBox)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.okPictureBox)).EndInit();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.GroupBox processPriorityGroupBox;
        private System.Windows.Forms.GroupBox processGroupBox;
        private System.Windows.Forms.ProgressBar processProgressBar;
        private System.Windows.Forms.GroupBox fileAnalyzeStatGroupBox;
        private System.Windows.Forms.Label percOfDamageLabel_;
        private System.Windows.Forms.Label percOfAltEccLabel_;
        private System.Windows.Forms.GroupBox logGroupBox;
        private System.Windows.Forms.GroupBox countGroupBox;
        private System.Windows.Forms.Label errorCountLabel_;
        private System.Windows.Forms.Label okCountLabel_;
        private System.Windows.Forms.ListBox logListBox;
        private System.Windows.Forms.ComboBox processPriorityComboBox;
        private System.Windows.Forms.PictureBox errorPictureBox;
        private System.Windows.Forms.PictureBox okPictureBox;
```

```
        private System.Windows.Forms.Label errorCountLabel;
        private System.Windows.Forms.Label okCountLabel;
        private System.Windows.Forms.ToolTip toolTip;
        private System.Windows.Forms.Timer closingTimer;
        private System.Windows.Forms.Label percOfAltEccLabel;
        private System.Windows.Forms.Label percOfDamageLabel;
        private PinkieControls.ButtonXP pauseButtonXP;
        private PinkieControls.ButtonXP stopButtonXP;
        private System.Windows.Forms.Timer processTimer;
    }
}
```

Файл FileAnalyzer.cs - контроль цілісності даних

```
using System;
using System.Threading;
using System.IO;

namespace Data_Center
{
    /// <summary>
    /// Клас контролю цілісності набору файлів-томів
    /// </summary>
    public class FileAnalyzer
    {
        #region Delegates

        /// <summary>
        /// Делегат відновлення прогресу контролю цілісності файлів
        /// </summary>
        public OnUpdateDoubleValueHandler OnUpdateFileAnalyzeProgress;

        /// <summary>
        /// Делегат завершення процесу контролю цілісності файлів
        /// </summary>
        public OnEventHandler OnFileAnalyzeFinish;

        /// <summary>
        /// Делегат одержання статистики ушкоджень багатотомного архіву
        /// </summary>
        public OnUpdateTwoDoubleValueHandler OnGetDamageStat;

        #endregion Delegates

        #region Public Properties & Data

        /// <summary>
        /// Булевська властивість "Файл обробляється?"
        /// </summary>
        public bool InProcessing
        {
            get
            {
                if (
                        (this.thrFileAnalyzer != null)
                    &&
                        (
                            (this.thrFileAnalyzer.ThreadState ==
ThreadState.Running)
                        ||
                            (this.thrFileAnalyzer.ThreadState ==
ThreadState.WaitSleepJoin)
                        )
                    )
                {
                    return true;

                } else
                {
                    return false;
                }
            }
        }

        /// <summary>
        /// Булевська властивість "Екземпляр класу закінчив обробку
        /// (має актуальний стан змінних-членів)?"
        /// </summary>
```

```
public bool Finished
{
    get
    {
        // Якщо клас не зайнятий обробкою - повертаємо значення
        if (!InProcessing)
        {
            return this.finished;

        } else
        {
            return false;
        }
    }
}

/// <summary>
/// Екземпляр класу повністю закінчив обробку?
/// </summary>
private bool finished;

/// <summary>
/// Булевська властивість "Безліч файлів оброблена коректно?"
/// </summary>
public bool ProcessedOK
{
    get
    {
        // Якщо клас не зайнятий обробкою - повертаємо значення
        if (!InProcessing)
        {
            return this.processedOK;

        } else
        {
            return false;
        }
    }
}

/// <summary>
/// Обробка набору файлів зроблена коректно?
/// </summary>
private bool processedOK;

/// <summary>
/// Список порядкових номерів наявних томів
/// </summary>
public int[] VolList
{
    get
    {
        if (!InProcessing)
        {
            return this.volList;

        } else
        {
            return null;
        }
    }
}

/// <summary>
/// Вектор, що вказує на состав томів
/// </summary>
private int[] volList;

/// <summary>
```

```
        /// Всі томи для відновлення коректні?
        /// </summary>
        public bool AllEccVolsOK
        {
            get
            {
                if (!InProcessing)
                {
                    return this.allEccVolsOK;

                } else
                {
                    return false;
                }
            }
        }

        /// <summary>
        /// Всі томи для відновлення коректні?
        /// </summary>
        private bool allEccVolsOK;

        /// <summary>
        /// Пріоритет процесу
        /// </summary>
        public int ThreadPriority
        {
            get
            {
                return (int)this.threadPriority;
            }

            set
            {
                if (
                        (this.thrFileAnalyzer != null)
                     &&
                        (this.thrFileAnalyzer.IsAlive)
                    )
                {
                    switch (value)
                    {
                        default:
                        case 0:
                        {
                            this.threadPriority =
System.Threading.ThreadPriority.Lowest;

                            break;
                        }

                        case 1:
                        {
                            this.threadPriority =
System.Threading.ThreadPriority.BelowNormal;

                            break;
                        }

                        case 2:
                        {
                            this.threadPriority =
System.Threading.ThreadPriority.Normal;

                            break;
                        }

                        case 3:
                        {
```

```
                            this.threadPriority =
System.Threading.ThreadPriority.AboveNormal;

                            break;
                        }

                        case 4:
                        {
                            this.threadPriority =
System.Threading.ThreadPriority.Highest;

                            break;
                        }
                    }

                    // Установлюємо обраний пріоритет процесу
                    this.thrFileAnalyzer.Priority = this.threadPriority;

                    // Дублюємо установку параметра для підконтрольного об'єкта
                    if (this.eFileIntegrityCheck != null)
                    {
                        this.eFileIntegrityCheck.ThreadPriority = value;
                    }
                }
            }
        }

        /// <summary>
        /// Пріоритет процесу контролю цілісності файлів
        /// </summary>
        private ThreadPriority threadPriority;

        /// <summary>
        /// Подія, установлювана по завершенню обробки
        /// </summary>
        public ManualResetEvent[] FinishedEvent
        {
            get
            {
                return this.finishedEvent;
            }
        }

        /// <summary>
        /// Подія, установлювана по завершенню обробки
        /// </summary>
        private ManualResetEvent[] finishedEvent;

        #endregion Public Properties & Data

        #region Data

        /// <summary>
        /// Модуль для впакування (розпакування) ім'я файлу в префіксний формат
        /// </summary>
        private FileNamer eFileNamer;

        /// <summary>
        /// Екземпляр класу контролю цілісності набору файлів
        /// </summary>
        private FileIntegrityCheck eFileIntegrityCheck;

        /// <summary>
        /// Шлях до файлів для обробки
        /// </summary>
        private String path;

        /// <summary>
        /// Ім'я файлу, якому належить безліч томів
```

```
        /// </summary>
        private String fileName;

        /// <summary>
        /// Кількість основних томів
        /// </summary>
        private int dataCount;

        /// <summary>
        /// Кількість томів для відновлення
        /// </summary>
        private int eccCount;

        /// <summary>
        /// Тип кодера коду Ріда-Соломона (по типу використовуваної матриці
кодування)
        /// </summary>
        private int codecType;

        /// <summary>
        /// Використовується швидке добування з томів (без перевірки CRC-64)?
        /// </summary>
        private bool fastExtraction;

        /// <summary>
        /// Потік контролю цілісності файлу
        /// </summary>
        private Thread thrFileAnalyzer;

        /// <summary>
        /// Подія припинення обробки файлів
        /// </summary>
        private ManualResetEvent[] exitEvent;

        /// <summary>
        /// Подія продовження обробки файлів
        /// </summary>
        private ManualResetEvent[] executeEvent;

        /// <summary>
        /// Подія "пробудження" циклу очікування
        /// </summary>
        private ManualResetEvent[] wakeUpEvent;

        #endregion Data

        #region Construction & Destruction

        /// <summary>
        /// Конструктор класу перевірки цілісності набору файлів
        /// </summary>
        public FileAnalyzer()
        {
            // Модуль для впакування (розпакування) ім'я файлу в префіксний
формат
            this.eFileNamer = new FileNamer();

            // Створюємо екземпляр класу контролю цілісності набору файлів
            this.eFileIntegrityCheck = new FileIntegrityCheck();

            // Шлях до файлів для обробки за замовчуванням порожній
            this.path = "";

            // Ініціалізуємо ім'я файлу за замовчуванням
            this.fileName = "NONAME";

            // Спочатку всі томи для відновлення вважаємо ушкодженими
            this.allEccVolsOK = false;
```

```
            // Екземпляр класу повністю закінчив обробку?
            this.finished = true;

            // Обробка зроблена коректно?
            this.processedOK = false;

            // За замовчуванням встановлюється фоновий пріоритет
            this.threadPriority = 0;

            // Ініціалізуємо подію припинення обробки файлів
            this.exitEvent = new ManualResetEvent[] { new
ManualResetEvent(false) };

            // Ініціалізуємо подію продовження обробки файлів
            this.executeEvent = new ManualResetEvent[] { new
ManualResetEvent(false) };

            // Ініціалізуємо подію "пробудження" циклу очікування
            this.wakeUpEvent = new ManualResetEvent[] { new
ManualResetEvent(false) };

            // Подія, установлювана по завершенню обробки
            this.finishedEvent = new ManualResetEvent[] { new
ManualResetEvent(true) };
        }

        #endregion Construction & Destruction

        #region Public Operations

        /// <summary>
        /// Метод запуску потоку обробки обчислення й записи CRC64 у кінець
файлів
        /// </summary>
        /// <param name="path">Шлях до файлів для обробки</param>
        /// <param name="fileName">Ім'я файлу для обробки</param>
        /// <param name="dataCount">Конфігурація кількості основних
томів</param>
        /// <param name="eccCount">Конфігурація кількості томів для
відновлення</param>
        /// <param name="codecType">Тип кодера коду Ріда-Соломона (по типу
матриці)</param>
        /// <param name="runAsSeparateThread">Запускати в окремому
потоці?</param>
        /// <returns>Булевський прапор операції</returns>
        public bool StartToWriteCRC64(String path, String fileName, int
dataCount, int eccCount, int codecType, bool runAsSeparateThread)
        {
            // Якщо потік обчислення CRC-64 працює - не дозволяємо повторний
запуск
            if (InProcessing)
            {
                return false;
            }

            // Скидаємо прапор коректності результату перед запуском потоку
            this.processedOK = false;

            // Скидаємо індикатор актуального стану змінних-членів
            this.finished = false;

            // Зберігаємо шлях до файлів для обробки
            if (path == null)
            {
                this.path = "";

            } else
            {
                // Робимо виділення шляху з "path" у випадку,
```

```
                // якщо туди було записано повне ім'я
                this.path = this.eFileNamer.GetPath(path);
            }

            if (fileName == null)
            {
                // Активуємо індикатор актуального стану змінних-членів
                this.finished = true;

                // Установлюємо подію завершення обробки
                this.finishedEvent[0].Set();

                return false;
            }

            // Робимо виділення короткого ім'я файлу з "fileName" у випадку,
            // якщо туди було записано повне ім'я
            this.fileName = this.eFileNamer.GetShortFileName(fileName);

            // Перевіряємо на некоректну конфігурацію
            if (
                    (dataCount <= 0)
                 ||
                    (eccCount <= 0)
                 ||
                    ((dataCount + eccCount) >
(int)Code_Rid_Solomon_Const.MaxVolCountAlt)
                )
            {
                // Активуємо індикатор актуального стану змінних-членів
                this.finished = true;

                // Установлюємо подію завершення обробки
                this.finishedEvent[0].Set();

                return false;
            }

            // Зберігаємо кількість основних томів
            this.dataCount = dataCount;

            // Зберігаємо кількість томів для відновлення
            this.eccCount = eccCount;

            // Зберігаємо тип кодера коду Ріда-Соломона (по типу
використовуваної матриці кодування)
            this.codecType = codecType;

            // Указуємо, що потік повинен виконуватися
            this.exitEvent[0].Reset();
            this.executeEvent[0].Set();
            this.wakeUpEvent[0].Reset();
            this.finishedEvent[0].Reset();

            // Якщо зазначено, що не потрібен запуск в окремому потоці,
            // запускаємо в даному
            if (!runAsSeparateThread)
            {
                // Обчислюємо CRC-64 для кожного з файлів набору
                WriteCRC64();

                // Повертаємо результат обробки
                return this.processedOK;
            }

            // Створюємо потік обчислення й запису CRC-64...
            this.thrFileAnalyzer = new Thread(new ThreadStart(WriteCRC64));

            //...потім даємо йому ім'я...
```

```
            this.thrFileAnalyzer.Name = "FileAnalyzer.WriteCRC64()";

            //...установлюємо обраний пріоритет завдання...
            this.thrFileAnalyzer.Priority = this.threadPriority;

            //...і запускаємо його
            this.thrFileAnalyzer.Start();

            // Повідомляємо, що все нормально
            return true;
        }

        /// <summary>
        /// Метод запуску потоку обробки перевірки CRC64, записаного в кінець
        /// кожного з файлів набору, з генеруванням списку наявних томів
"volList",
        /// який буде використаний декодером для відновлення даних
        /// </summary>
        /// <param name="path">Шлях до файлів для обробки</param>
        /// <param name="fileName">Ім'я файлу для обробки</param>
        /// <param name="dataCount">Конфігурація кількості основних
томів</param>
        /// <param name="eccCount">Конфігурація кількості томів для
відновлення</param>
        /// <param name="codecType">Тип кодера коду Ріда-Соломона (по типу
матриці)</param>
        /// <param name="fastExtraction">Використовується швидке добування з
томів (без перевірки CRC-64)?</param>
        /// <param name="runAsSeparateThread">Запускати в окремому
потоці?</param>
        /// <returns>Булевський прапор операції</returns>
        public bool StartToAnalyzeCRC64(String path, String fileName, int
dataCount, int eccCount, int codecType, bool fastExtraction, bool
runAsSeparateThread)
        {
            // Якщо потік обчислення CRC-64 працює - не дозволяємо повторний
запуск
            if (InProcessing)
            {
                return false;
            }

            // Спочатку всі томи для відновлення вважаємо ушкодженими
            this.allEccVolsOK = false;

            // Скидаємо прапор коректності результату перед запуском потоку
            this.processedOK = false;

            // Скидаємо індикатор актуального стану змінних-членів
            this.finished = false;

            // Зберігаємо шлях до файлів для обробки
            if (path == null)
            {
                this.path = "";

            } else
            {
                // Робимо виділення шляху з "path" у випадку,
                // якщо туди було записано повне ім'я
                this.path = this.eFileNamer.GetPath(path);
            }

            if (fileName == null)
            {
                // Активуємо індикатор актуального стану змінних-членів
                this.finished = true;

                // Установлюємо подію завершення обробки
```

```
                this.finishedEvent[0].Set();

                return false;
            }

            // Робимо виділення короткого ім'я файлу з "fileName" у випадку,
            // якщо туди було записано повне ім'я
            this.fileName = this.eFileNamer.GetShortFileName(fileName);

            // Перевіряємо на некоректну конфігурацію
            if (
                    (dataCount <= 0)
                 ||
                    (eccCount <= 0)
                 ||
                    ((dataCount + eccCount) >
(int)Code_Rid_Solomon_Const.MaxVolCountAlt)
                )
            {
                // Активуємо індикатор актуального стану змінних-членів
                this.finished = true;

                // Установлюємо подію завершення обробки
                this.finishedEvent[0].Set();

                return false;
            }

            // Зберігаємо кількість основних томів
            this.dataCount = dataCount;

            // Зберігаємо кількість томів для відновлення
            this.eccCount = eccCount;

            // Зберігаємо тип кодека кодера коду Ріда-Соломона (по типу
використовуваної матриці кодування)
            this.codecType = codecType;

            // Використовується швидке добування з томів (без перевірки CRC-64)?
            this.fastExtraction = fastExtraction;

            // Указуємо, що потік повинен виконуватися
            this.exitEvent[0].Reset();
            this.executeEvent[0].Set();
            this.wakeUpEvent[0].Reset();
            this.finishedEvent[0].Reset();

            // Якщо зазначено, що не потрібен запуск в окремому потоці,
            // запускаємо в даному
            if (!runAsSeparateThread)
            {
                // Обчислюємо й перевіряємо CRC-64 для кожного з файлів набору
із заповненням
                // властивості VolList
                AnalyzeCRC64();

                // Повертаємо результат обробки
                return this.processedOK;
            }

            // Створюємо потік обчислення й перевірки CRC-64...
            this.thrFileAnalyzer = new Thread(new ThreadStart(AnalyzeCRC64));

            //...потім даємо йому ім'я...
            this.thrFileAnalyzer.Name = "FileAnalyzer.AnalyzeCRC64()";

            //...установлюємо обраний пріоритет завдання...
            this.thrFileAnalyzer.Priority = this.threadPriority;
```

```
            //...і запускаємо його
            this.thrFileAnalyzer.Start();

            // Повідомляємо, що все нормально
            return true;
        }

        /// <summary>
        /// Метод зупинки потоку
        /// </summary>
        public void Stop()
        {
            // Указуємо, що потік обробки більше не повинен виконуватися
            this.exitEvent[0].Set();

            // Примусово знімаємо з паузи
            this.executeEvent[0].Set();

            // Знімаємо з очікування в циклі
            this.wakeUpEvent[0].Set();
        }

        /// <summary>
        /// Постановка потоку обробки на паузу
        /// </summary>
        public void Pause()
        {
            // Ставимо на паузу
            this.executeEvent[0].Reset();

            // Знімаємо з очікування в циклі
            this.wakeUpEvent[0].Set();
        }

        /// <summary>
        /// Зняття потоку обробки з паузи
        /// </summary>
        public void Continue()
        {
            // Знімаємо обробку с паузи
            this.executeEvent[0].Set();
        }

        #endregion Public Operations

        #region Private Operations

        /// <summary>
        /// Обчислення й запис у кінець файлів значення CRC-64
        /// </summary>
        private void WriteCRC64()
        {
            // Обчислюємо значення модуля, що дозволить виводити відсоток
обробки
            // рівно при одиничному збільшенні для циклу по "i"
            int progressMod1 = (this.dataCount + this.eccCount) / 100;

            // Якщо модуль дорівнює нулю, то збільшуємо його до значення "1",
щоб
            // прогрес виводився на кожній ітерації (файл дуже маленький)
            if (progressMod1 == 0)
            {
                progressMod1 = 1;
            }

            // Піддаємо обробці всі томи
            for (int volNum = 0; volNum < (this.dataCount + this.eccCount);
volNum++)
            {
```

```
                // Зчитуємо первісне ім'я файлу
                String fileName = this.fileName;

                // Одержуємо ім'я вихідного файлу в префіксній формі
                this.eFileNamer.Pack(ref fileName, volNum, this.dataCount,
this.eccCount, this.codecType);

                // Формуємо повне ім'я файлу
                fileName = this.path + fileName;

                // Робимо обчислення CRC-64 для кожного файлу
                if (this.eFileIntegrityCheck.StartToWriteCRC64(fileName, true))
                {
                    // Цикл очікування завершення обробки файлу
                    while (true)
                    {
                        // Якщо не виявили встановленої події "executeEvent",
                        // те користувач хоче, щоб ми поставили обробку на паузу
-
                        if (!ManualResetEvent.WaitAll(this.executeEvent, 0,
false))
                        {
                            //...припиняємо роботу контрольованого алгоритму...
                            this.eFileIntegrityCheck.Pause();

                            //...програма переходить у режим сна
                            ManualResetEvent.WaitAll(this.executeEvent);

                            // А коли прокинулися, указуємо, що обробка повинна
тривати
                            this.eFileIntegrityCheck.Continue();
                        }

                        // Чекаємо кожне з перерахованих подій...
                        int eventIdx = ManualResetEvent.WaitAny(new
ManualResetEvent[] { this.wakeUpEvent[0], this.exitEvent[0],
this.eFileIntegrityCheck.FinishedEvent[0] });

                        //...якщо одержали сигнал до того, щоб прокинутися -
                        // переходимо на нову ітерацію, тому що прокидаємося
                        // перед постановкою на паузу...
                        if (eventIdx == 0)
                        {
                            //...попередньо скинувши подію, що змусила нас
прокинутися
                            this.wakeUpEvent[0].Reset();

                            continue;
                        }

                        //...якщо одержали сигнал до виходу з обробки...
                        if (eventIdx == 1)
                        {
                            //...зупиняємо контрольований алгоритм
                            this.eFileIntegrityCheck.Stop();

                            // Указуємо на те, що обробка була перервана
                            this.processedOK = false;

                            // Активуємо індикатор актуального стану змінних-
членів
                            this.finished = true;

                            // Установлюємо подію завершення обробки
                            this.finishedEvent[0].Set();

                            return;
                        }
```

```
                        //...якщо одержали сигнал про завершення обробки
вкладеним алгоритмом...
                        if (eventIdx == 2)
                        {
                            //...виходимо із циклу очікування завершення (цього
й чекали в while(true)!)
                            break;
                        }

                    } // while(true)

                } else
                {
                    // Скидаємо прапор коректності результату
                    this.processedOK = false;

                    // Активуємо індикатор актуального стану змінних-членів
                    this.finished = true;

                    // Установлюємо подію завершення обробки
                    this.finishedEvent[0].Set();

                    return;
                }

                // У зв'язку із закриттям великої кількості файлових потоків
                // необхідно дочекатися запису змін, внесених потоком
                // кодування в тіло класу. Потік уже не працює, але
                // установлена їм Булевська властивість, можливо, ще
                // "не виявилося"
                for (int i = 0; i < (int)WaitCount.MaxWaitCount; i++)
                {
                    if (!this.eFileIntegrityCheck.Finished)
                    {
                        Thread.Sleep((int)WaitTime.MinWaitTime);

                    } else
                    {
                        break;
                    }
                }

                // Якщо цикли очікування закриття файлових потоків не привели до
бажаного
                // результату - це помилка
                if (!this.eFileIntegrityCheck.ProcessedOK)
                {
                    // Указуємо на те, що обробка не була завершена коректно
                    this.processedOK = false;

                    // Активуємо індикатор актуального стану змінних-членів
                    this.finished = true;

                    // Установлюємо подію завершення обробки
                    this.finishedEvent[0].Set();

                    return;
                }

                // Виводимо прогрес обробки
                if (
                        ((volNum % progressMod1) == 0)
                     &&
                        (OnUpdateFileAnalyzeProgress != null)
                    )
                {
                    OnUpdateFileAnalyzeProgress(((double)(volNum + 1) /
(double)(this.dataCount + this.eccCount)) * 100.0);
                }
```

```
                // У випадку, якщо потрібна постановка на паузу, подію
"executeEvent"
                // буде скинуто, і будемо на паузі аж до його появи
                ManualResetEvent.WaitAll(this.executeEvent);

                // Якщо зазначено, що потрібно вийти з потоку - виходимо
                if (ManualResetEvent.WaitAll(this.exitEvent, 0, false))
                {
                    // Указуємо на те, що обробка була перервана
                    this.processedOK = false;

                    // Активуємо індикатор актуального стану змінних-членів
                    this.finished = true;

                    // Установлюємо подію завершення обробки
                    this.finishedEvent[0].Set();

                    return;
                }
            }

            // Повідомляємо про закінчення процесу обробки
            if (OnFileAnalyzeFinish != null)
            {
                OnFileAnalyzeFinish();
            }

            // Повідомляємо, що обробка пройшла коректно
            this.processedOK = true;

            // Активуємо індикатор актуального стану змінних-членів
            this.finished = true;

            // Установлюємо подію завершення обробки
            this.finishedEvent[0].Set();
        }

          /// <summary>
        /// Обчислення й перевірка значення CRC-64, записаного наприкінці файлу
        /// </summary>
        private void AnalyzeCRC64()
        {
            // Обчислюємо значення модуля, що дозволить виводити відсоток
обробки
            // рівно при одиничному збільшенні для циклу по "i"
            int progressMod1 = (this.dataCount + this.eccCount) / 100;

            // Якщо модуль дорівнює нулю, то збільшуємо його до значення "1",
щоб
            // прогрес виводився на кожній ітерації (файл дуже маленький)
            if (progressMod1 == 0)
            {
                progressMod1 = 1;
            }

            // Виділяємо пам'ять під "volList"
            this.volList = new int[this.dataCount];

            // Виділяємо пам'ять під "altEccList"
            int[] altEccList = new int[this.eccCount];

            // Індекс у масиві томів
            int volListIdx = 0;

            // Індекс у масиві томів для відновлення
            int altEccListIdx = 0;

            // Лічильник кількості ушкоджених основних томів
```

```
            int dataVolMissCount = 0;

            // Лічильник кількості знайдених томів для відновлення
            int eccVolPresentCount = 0;

            // Ім'я файлу для обробки
            String fileName;

            // Піддаємо перевірці всі основні томи
            for (int dataNum = 0; dataNum < this.dataCount; dataNum++)
            {
                // Спочатку припускаємо, що поточний том ушкоджено
                bool dataVolIsOK = false;

                // Зчитуємо первісне ім'я файлу
                fileName = this.fileName;

                // Одержуємо ім'я вихідного файлу в префіксній формі
                this.eFileNamer.Pack(ref fileName, dataNum, this.dataCount,
this.eccCount, this.codecType);

                // Формуємо повне ім'я файлу
                fileName = this.path + fileName;

                // Якщо вихідний файл існує...
                if (File.Exists(fileName))
                {
                    // Якщо не використовується швидке добування - перевіряємо
на цілісність
                    // CRC-64, інакше думаємо, що все корректно (цілісність тому
беремо
                    // по факті його наявності)
                    if (!this.fastExtraction)
                    {
                        //...- робимо його перевірку
                        if (this.eFileIntegrityCheck.StartToCheckCRC64(fileName,
true))
                        {
                            // Цикл очікування завершення обробки файлу
                            while (true)
                            {
                                // Якщо не виявили встановленої події
"executeEvent",
                                // те користувач хоче, щоб ми поставили обробку
на паузу -
                                if (!ManualResetEvent.WaitAll(this.executeEvent,
0, false))
                                {
                                    //...припиняємо роботу контрольованого
алгоритму...
                                    this.eFileIntegrityCheck.Pause();

                                    //...програма переходить у режим сна
                                    ManualResetEvent.WaitAll(this.executeEvent);

                                    // А коли прокинулися, указуємо, що обробка
повинна тривати
                                    this.eFileIntegrityCheck.Continue();
                                }

                                // Чекаємо кожне з перерахованих подій...
                                int eventIdx = ManualResetEvent.WaitAny(new
ManualResetEvent[] { this.wakeUpEvent[0], this.exitEvent[0],
this.eFileIntegrityCheck.FinishedEvent[0] });

                                //...якщо одержали сигнал до того, щоб
прокинутися -
                                // переходимо на нову ітерацію, тому що
прокидаємося
```

```
                                // перед постановкою на паузу...
                                if (eventIdx == 0)
                                {
                                    //...попередньо скинувши подію, що змусила
нас прокинутися
                                    this.wakeUpEvent[0].Reset();

                                    continue;
                                }

                                //...якщо одержали сигнал до виходу з обробки...
                                if (eventIdx == 1)
                                {
                                    //...зупиняємо контрольований алгоритм
                                    this.eFileIntegrityCheck.Stop();

                                    // Указуємо на те, що обробка була перервана
                                    this.processedOK = false;

                                    // Активуємо індикатор актуального стану
змінних-членів
                                    this.finished = true;

                                    // Установлюємо подію завершення обробки
                                    this.finishedEvent[0].Set();

                                    return;
                                }

                                //...якщо одержали сигнал про завершення обробки
вкладеним алгоритмом...
                                if (eventIdx == 2)
                                {
                                    //...виходимо із циклу очікування завершення
(цього й чекали в while(true)!)
                                    break;
                                }

                            } // while(true)

                        } else
                        {
                            // Скидаємо прапор коректності результату
                            this.processedOK = false;

                            // Активуємо індикатор актуального стану змінних-
членів
                            this.finished = true;

                            // Установлюємо подію завершення обробки
                            this.finishedEvent[0].Set();

                            return;
                        }

                        // У зв'язку із закриттям великої кількості файлових
потоків
                        // необхідно дочекатися запису змін, внесених потоком
                        // кодування в тіло класу. Потік уже не працює, але
                        // установлене їм Булевська властивість, можливо, ще
                        // "не виявилося"
                        for (int i = 0; i < (int)WaitCount.MaxWaitCount; i++)
                        {
                            if (!this.eFileIntegrityCheck.Finished)
                            {
                                Thread.Sleep((int)WaitTime.MinWaitTime);

                            } else
                            {
```

```
                                break;
                            }
                        }

                        // Указуємо, що основний том коректний
                        if (this.eFileIntegrityCheck.ProcessedOK)
                        {
                            dataVolIsOK = true;
                        }

                    } else
                    {
                        // Указуємо, що основний том коректний
                        dataVolIsOK = true;
                    }

                    // Виводимо прогрес обробки
                    if (
                            ((dataNum % progressMod1) == 0)
                     &&
                            (OnUpdateFileAnalyzeProgress != null)
                        )
                    {
                        OnUpdateFileAnalyzeProgress(((double)(dataNum + 1) /
(double)(this.dataCount + this.eccCount)) * 100.0);
                    }

                    // У випадку, якщо потрібна постановка на паузу, подію
"executeEvent"
                    // буде скинуто, і будемо на паузі аж до його появи
                    ManualResetEvent.WaitAll(this.executeEvent);

                    // Якщо зазначено, що потрібно вийти з потоку - виходимо
                    if (ManualResetEvent.WaitAll(this.exitEvent, 0, false))
                    {
                        // Указуємо на те, що обробка була перервана
                        this.processedOK = false;

                        // Активуємо індикатор актуального стану змінних-членів
                        this.finished = true;

                        // Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();

                        return;
                    }
                }

                // Якщо даний основний том не ушкоджений, записуємо його в
"volList",
                // а інакше збільшуємо лічильник ушкоджених томів і ставимо на
місце
                // номера тому значення "-1", що вкаже на необхідність
підстановки
                // тому для відновлення
                if (dataVolIsOK)
                {
                    this.volList[volListIdx++] = dataNum;

                } else
                {
                    this.volList[volListIdx++] = -1;

                    // Збільшуємо лічильник кількості ушкоджених основних томів
                    dataVolMissCount++;
                }
            }

            // Тепер, коли знаємо кількість ушкоджених основних томів,
```

```
            // потрібно просканувати всі файли для відновлення, і визначити
            // необхідну їхню частину в список томів, а "надлишок" помістити в
            // список альтренативних томів для відновлення
            for (int eccNum = this.dataCount; eccNum < (this.dataCount +
this.eccCount); eccNum++)
            {
                // Спочатку припускаємо, що поточний том ушкоджено
                bool eccVolIsOK = false;

                // Зчитуємо первісне ім'я файлу
                fileName = this.fileName;

                // Одержуємо ім'я вихідного файлу в префіксній формі
                this.eFileNamer.Pack(ref fileName, eccNum, this.dataCount,
this.eccCount, this.codecType);

                // Формуємо повне ім'я файлу
                fileName = this.path + fileName;

                // Якщо вихідний файл існує...
                if (File.Exists(fileName))
                {
                    // Якщо не використовується швидке добування - перевіряємо
на цілісність
                    // CRC-64, інакше думаємо, що все корректно (цілісність тому
беремо
                    // по факту його наявності)
                    if (!this.fastExtraction)
                    {
                        //...- робимо його перевірку
                        if (this.eFileIntegrityCheck.StartToCheckCRC64(fileName,
true))
                        {
                            // Цикл очікування завершення обробки файлу
                            while (true)
                            {
                                // Якщо не виявили встановленої події
"executeEvent",
                                // то користувач хоче, щоб ми поставили обробку
на паузу -
                                if (!ManualResetEvent.WaitAll(this.executeEvent,
0, false))
                                {
                                    //...припиняємо роботу контрольованого
алгоритму...
                                    this.eFileIntegrityCheck.Pause();

                                    //...програма переходить у режим сна
                                    ManualResetEvent.WaitAll(this.executeEvent);

                                    // А коли прокинулися, указуємо, що обробка
повинна тривати
                                    this.eFileIntegrityCheck.Continue();
                                }

                                // Чекаємо кожне з перерахованих подій...
                                int eventIdx = ManualResetEvent.WaitAny(new
ManualResetEvent[] { this.wakeUpEvent[0], this.exitEvent[0],
this.eFileIntegrityCheck.FinishedEvent[0] });

                                //...якщо одержали сигнал до того, щоб
прокинутися -
                                // переходимо на нову ітерацію, тому що
прокидаємося
                                // перед постановкою на паузу...
                                if (eventIdx == 0)
                                {
                                    //...попередньо скинувши подію, що змусила
нас прокинутися
```

```
                                this.wakeUpEvent[0].Reset();

                                continue;
                            }

                            //...якщо одержали сигнал до виходу з обробки...
                            if (eventIdx == 1)
                            {
                                //...зупиняємо контрольований алгоритм
                                this.eFileIntegrityCheck.Stop();

                                // Указуємо на те, що обробка була перервана
                                this.processedOK = false;

                                // Активуємо індикатор актуального стану
змінних-членів
                                this.finished = true;

                                // Установлюємо подію завершення обробки
                                this.finishedEvent[0].Set();

                                return;
                            }

                            //...якщо одержали сигнал про завершення обробки
вкладеним алгоритмом...
                            if (eventIdx == 2)
                            {
                                //...виходимо із циклу очікування завершення
(цього й чекали в while(true)!)
                                break;
                            }

                        } // while(true)

                    } else
                    {
                        // Скидаємо прапор коректності результату
                        this.processedOK = false;

                        // Активуємо індикатор актуального стану змінних-
членів
                        this.finished = true;

                        // Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();

                        return;
                    }

                    // У зв'язку із закриттям великої кількості файлових
потоків
                    // необхідно дочекатися запису змін, внесених потоком
                    // кодування в тіло класу. Потік уже не працює, але
                    // установлена їм Булевська властивість, можливо, ще
                    // "не виявилася"
                    for (int i = 0; i < (int)WaitCount.MaxWaitCount; i++)
                    {
                        if (!this.eFileIntegrityCheck.Finished)
                        {
                            Thread.Sleep((int)WaitTime.MinWaitTime);

                        } else
                        {
                            break;
                        }
                    }

                    // Указуємо, що том для відновлення коректний
```

```
                        if (this.eFileIntegrityCheck.ProcessedOK)
                        {
                            eccVolIsOK = true;
                        }

                    } else
                    {
                        // Указуємо, що том для відновлення коректний
                        eccVolIsOK = true;
                    }

                    // Виводимо прогрес обробки
                    if (
                            ((eccNum % progressMod1) == 0)
                         &&
                            (OnUpdateFileAnalyzeProgress != null)
                        )
                    {
                        OnUpdateFileAnalyzeProgress(((double)(eccNum + 1) /
(double)(this.dataCount + this.eccCount)) * 100.0);
                    }

                    // У випадку, якщо потрібна постановка на паузу, подію
"executeEvent"
                    // буде скинуто, і будемо на паузі аж до його появи
                    ManualResetEvent.WaitAll(this.executeEvent);

                    // Якщо зазначено, що потрібно вийти з потоку - виходимо
                    if (ManualResetEvent.WaitAll(this.exitEvent, 0, false))
                    {
                        // Указуємо на те, що обробка була перервана
                        this.processedOK = false;

                        // Активуємо індикатор актуального стану змінних-членів
                        this.finished = true;

                        // Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();

                        return;
                    }
                }

                // Якщо том для відновлення гарний...
                if (eccVolIsOK)
                {
                    //...- додаємо його в список
                    altEccList[altEccListIdx++] = eccNum;

                    // Збільшуємо лічильник кількості томів для відновлення
                    eccVolPresentCount++;

                } else
                {
                    //...а інакше вказуємо, що том ушкоджено
                    altEccList[altEccListIdx++] = -1;
                }
            }

            // Якщо значення лічильника кількості коректних томів для
відновлення збігається
            // зі значенням лічильника томів для відновлення конфігурації - всі
томи для
            // відновлення є неушкодженими
            if (eccVolPresentCount == this.eccCount)
            {
                this.allEccVolsOK = true;
            }
```

```
            // Виводимо статистику ушкоджень
            if (OnGetDamageStat != null)
            {
                // Обчислюємо загальний відсоток ушкоджень (суму ушкоджень
                // основних томів і томів для відновлення ділимо на загальну
кількість томів)
                double percOfDamage = ((double)(dataVolMissCount +
(this.eccCount - eccVolPresentCount)) / (double)(this.dataCount +
this.eccCount)) * 100;

                // Обчислюємо відсоток "" альтернативних томів, щовижили, для
відновлення
                // Альтернативні томи - це спочатку ті томи, які не планується
використовувати для відновлення
                double percOfAltEcc = ((double)(eccVolPresentCount -
dataVolMissCount) / (double)this.eccCount) * 100;

                // Виводимо статистику ушкоджень
                OnGetDamageStat(percOfDamage, percOfAltEcc);
            }

            // Якщо немає ушкоджених основних томів, просто виходимо
            if (dataVolMissCount == 0)
            {
                // Повідомляємо про закінчення процесу обробки
                if (OnFileAnalyzeFinish != null)
                {
                    OnFileAnalyzeFinish();
                }

                // Указуємо на те, що дані не ушкоджені
                this.processedOK = true;

                // Активуємо індикатор актуального стану змінних-членів
                this.finished = true;

                // Установлюємо подію завершення обробки
                this.finishedEvent[0].Set();

                return;
            }

            // Якщо ми не зможемо відновити ушкодження...
            if (eccVolPresentCount < dataVolMissCount)
            {
                //...указуємо на те, що дані не можуть бути відновлені
                this.processedOK = false;

                // Активуємо індикатор актуального стану змінних-членів
                this.finished = true;

                // Установлюємо подію завершення обробки
                this.finishedEvent[0].Set();

                return;
            }

            // Переміщаємося на початок списку альтернативних томів для
відновлення
            altEccListIdx = 0;

            // Тепер пробігаємося по вектору "volList", і замість кожного зі
значень "-1"
            // підставляємо чергове значення зі знайденого діапазону
            for (int i = 0; i < this.dataCount; i++)
            {
                if (this.volList[i] == -1)
                {
                    // Пробігаємося по векторі томів для відновлення,
```

```
                        // зупиняючись на коректному томі для відновлення
                        while (altEccList[altEccListIdx] == -1)
                        {
                            altEccListIdx++;
                        }

                        // Підставляємо на місце ушкодженого основного тому
                        // том для відновлення,...
                        this.volList[i] = altEccList[altEccListIdx];

                        //...забираючи використаний том зі списку альтернативних
                        altEccList[altEccListIdx] = -1;
                    }
                }

                // Повідомляємо про закінчення процесу обробки
                if (OnFileAnalyzeFinish != null)
                {
                    OnFileAnalyzeFinish();
                }

                // Повідомляємо, що обробка пройшла коректно
                this.processedOK = true;

                // Активуємо індикатор актуального стану змінних-членів
                this.finished = true;

                // Установлюємо подію завершення обробки
                this.finishedEvent[0].Set();
            }

            #endregion Private Operations
        }
    }
```

Файл MainForm.cs - головне вікно програми

```
namespace Data_Center
{
    partial class MainForm
    {
        /// <summary>
        ///
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        ///
        /// </summary>
        /// <param name="disposing">правда, якщо керуючі ресурси повині бути
розташовані, неправда в іншому випадку.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Викликаємо метод для підтримки інтерфейсу
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.Windows.Forms.TreeNode treeNode1 = new
System.Windows.Forms.TreeNode("");
            System.Windows.Forms.TreeNode treeNode2 = new
System.Windows.Forms.TreeNode("Documents and Settings", 18, 20, new
System.Windows.Forms.TreeNode[] {
            treeNode1});
            System.Windows.Forms.TreeNode treeNode3 = new
System.Windows.Forms.TreeNode("");
            System.Windows.Forms.TreeNode treeNode4 = new
System.Windows.Forms.TreeNode("Завантаження", 18, 20, new
System.Windows.Forms.TreeNode[] {
            treeNode3});
            System.Windows.Forms.TreeNode treeNode5 = new
System.Windows.Forms.TreeNode("");
            System.Windows.Forms.TreeNode treeNode6 = new
System.Windows.Forms.TreeNode("Program Files", 18, 20, new
System.Windows.Forms.TreeNode[] {
            treeNode5});
            System.Windows.Forms.TreeNode treeNode7 = new
System.Windows.Forms.TreeNode("");
            System.Windows.Forms.TreeNode treeNode8 = new
System.Windows.Forms.TreeNode("RapidDriver", 18, 20, new
System.Windows.Forms.TreeNode[] {
            treeNode7});
            System.Windows.Forms.TreeNode treeNode9 = new
System.Windows.Forms.TreeNode("");
            System.Windows.Forms.TreeNode treeNode10 = new
System.Windows.Forms.TreeNode("Server", 18, 20, new
System.Windows.Forms.TreeNode[] {
            treeNode9});
            System.Windows.Forms.TreeNode treeNode11 = new
System.Windows.Forms.TreeNode("");
```

```
            System.Windows.Forms.TreeNode treeNode12 = new
System.Windows.Forms.TreeNode("WINDOWS", 18, 20, new
System.Windows.Forms.TreeNode[] {
            treeNode11});
            System.Windows.Forms.TreeNode treeNode13 = new
System.Windows.Forms.TreeNode("");
            System.Windows.Forms.TreeNode treeNode14 = new
System.Windows.Forms.TreeNode("WINDOWS.0", 18, 20, new
System.Windows.Forms.TreeNode[] {
            treeNode13});
            System.Windows.Forms.TreeNode treeNode15 = new
System.Windows.Forms.TreeNode("SYSTEM2 (C:)", 23, 24, new
System.Windows.Forms.TreeNode[] {
            treeNode2,
            treeNode4,
            treeNode6,
            treeNode8,
            treeNode10,
            treeNode12,
            treeNode14});
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(MainForm));
            this.menuStrip = new System.Windows.Forms.MenuStrip();
            this.fileToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.instrumentToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.adjustmentToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.encodingfilterToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.quickextractionToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.helpToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.separatorToolStripMenuItem = new
System.Windows.Forms.ToolStripSeparator();
            this.aboutToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.coderConfigGroupBox = new System.Windows.Forms.GroupBox();
            this.redundancyGroupBox = new System.Windows.Forms.GroupBox();
            this.redundancyMacTrackBar = new
EConTech.Windows.MACUI.MACTrackBar();
            this.allVolCountGroupBox = new System.Windows.Forms.GroupBox();
            this.allVolCountMacTrackBar = new
EConTech.Windows.MACUI.MACTrackBar();
            this.toolTip = new System.Windows.Forms.ToolTip(this.components);
            this.browser = new FileBrowser.Browser();
            this.repairButton = new System.Windows.Forms.Button();
            this.testButton = new System.Windows.Forms.Button();
            this.recoverButton = new System.Windows.Forms.Button();
            this.protectButton = new System.Windows.Forms.Button();
            this.exitToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.testspeedToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.menuStrip.SuspendLayout();
            this.coderConfigGroupBox.SuspendLayout();
            this.redundancyGroupBox.SuspendLayout();
            this.allVolCountGroupBox.SuspendLayout();
            this.SuspendLayout();
            //
            // menuStrip
            //
            this.menuStrip.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.fileToolStripMenuItem,
            this.instrumentToolStripMenuItem,
            this.adjustmentToolStripMenuItem,
```

```
            this.helpToolStripMenuItem});
            this.menuStrip.LayoutStyle =
System.Windows.Forms.ToolStripLayoutStyle.Flow;
            this.menuStrip.Location = new System.Drawing.Point(0, 0);
            this.menuStrip.Name = "menuStrip";
            this.menuStrip.Size = new System.Drawing.Size(986, 21);
            this.menuStrip.TabIndex = 0;
            this.menuStrip.Text = "menuStrip";
            //
            // fileToolStripMenuItem
            //
            this.fileToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.exitToolStripMenuItem});
            this.fileToolStripMenuItem.Name = "файлToolStripMenuItem";
            this.fileToolStripMenuItem.Size = new System.Drawing.Size(45, 17);
            this.fileToolStripMenuItem.Text = "Файл";
            //
            // instrumentsToolStripMenuItem
            //
            this.instrumentToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.testspeedToolStripMenuItem});
            this.instrumentToolStripMenuItem.Name =
"instrumentsToolStripMenuItem";
            this.instrumentToolStripMenuItem.Size = new System.Drawing.Size(82,
17);
            this.instrumentToolStripMenuItem.Text = "Інструменти";
            //
            // adjustmentToolStripMenuItem
            //
            this.adjustmentToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.encodingfilterToolStripMenuItem,
            this.quickextractionToolStripMenuItem});
            this.adjustmentToolStripMenuItem.Name =
"adjustmentToolStripMenuItem";
            this.adjustmentToolStripMenuItem.Size = new System.Drawing.Size(74,
17);
            this.adjustmentToolStripMenuItem.Text = "Параметри";
            //
            // encodingfilterToolStripMenuItem
            //
            this.encodingfilterToolStripMenuItem.ImageScaling =
System.Windows.Forms.ToolStripItemImageScaling.None;
            this.encodingfilterToolStripMenuItem.Name =
"encodingfilterToolStripMenuItem";
            this.encodingfilterToolStripMenuItem.Size = new
System.Drawing.Size(216, 22);
            this.encodingfilterToolStripMenuItem.Text = "Шифруючий фільтр";
            this.encodingfilterToolStripMenuItem.Click += new
System.EventHandler(this.encodingfilterToolStripMenuItem_Click);
            //
            // helpToolStripMenuItem
            //
            this.helpToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.separatorToolStripMenuItem,
            this.aboutToolStripMenuItem});
            this.helpToolStripMenuItem.Name = "helpToolStripMenuItem";
            this.helpToolStripMenuItem.Size = new System.Drawing.Size(60, 17);
            this.helpToolStripMenuItem.Text = "Довідка";
            //
            // separatorToolStripMenuItem
            //
            this.separatorToolStripMenuItem.Name = "separatorToolStripMenuItem";
            this.separatorToolStripMenuItem.Size = new System.Drawing.Size(163,
6);
            //
```

```
            // aboutToolStripMenuItem
            //
            this.aboutToolStripMenuItem.Name = "aboutToolStripMenuItem";
            this.aboutToolStripMenuItem.Size = new System.Drawing.Size(166, 22);
            this.aboutToolStripMenuItem.Text = "Про програму...";
            this.aboutToolStripMenuItem.Click += new
System.EventHandler(this.aboutToolStripMenuItem_Click);
            //
            // coderConfigGroupBox
            //
            this.coderConfigGroupBox.Controls.Add(this.redundancyGroupBox);
            this.coderConfigGroupBox.Controls.Add(this.allVolCountGroupBox);
            this.coderConfigGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.coderConfigGroupBox.Location = new System.Drawing.Point(414,
26);
            this.coderConfigGroupBox.Name = "coderConfigGroupBox";
            this.coderConfigGroupBox.Size = new System.Drawing.Size(561, 98);
            this.coderConfigGroupBox.TabIndex = 5;
            this.coderConfigGroupBox.TabStop = false;
            this.coderConfigGroupBox.Text = "Конфігурація системи";
            //
            // redundancyGroupBox
            //
            this.redundancyGroupBox.Controls.Add(this.redundancyMacTrackBar);
            this.redundancyGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.redundancyGroupBox.Location = new System.Drawing.Point(286,
21);
            this.redundancyGroupBox.Name = "redundancyGroupBox";
            this.redundancyGroupBox.Size = new System.Drawing.Size(264, 65);
            this.redundancyGroupBox.TabIndex = 4;
            this.redundancyGroupBox.TabStop = false;
            this.redundancyGroupBox.Text = "Надлишковість кодування";
            //
            // allVolCountGroupBox
            //
            this.allVolCountGroupBox.Controls.Add(this.allVolCountMacTrackBar);
            this.allVolCountGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.allVolCountGroupBox.Location = new System.Drawing.Point(12,
21);
            this.allVolCountGroupBox.Name = "allVolCountGroupBox";
            this.allVolCountGroupBox.Size = new System.Drawing.Size(264, 65);
            this.allVolCountGroupBox.TabIndex = 3;
            this.allVolCountGroupBox.TabStop = false;
            this.allVolCountGroupBox.Text = "Довжина коду";
            //
            // toolTip
            //
            this.toolTip.AutomaticDelay = 2000;
            this.toolTip.AutoPopDelay = 20000;
            this.toolTip.InitialDelay = 2000;
            this.toolTip.ReshowDelay = 1000;
            //
            // redundancyMacTrackBar
            //
            this.redundancyMacTrackBar.BackColor =
System.Drawing.Color.Transparent;
            this.redundancyMacTrackBar.BorderColor =
System.Drawing.SystemColors.ActiveBorder;
            this.redundancyMacTrackBar.Font = new System.Drawing.Font("Verdana",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.redundancyMacTrackBar.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(123)))), ((int)(((byte)(125)))),
((int)(((byte)(123)))));
            this.redundancyMacTrackBar.IndentHeight = 6;
```

```
            this.redundancyMacTrackBar.Location = new System.Drawing.Point(6,
24);
            this.redundancyMacTrackBar.Maximum = 199;
            this.redundancyMacTrackBar.Minimum = 0;
            this.redundancyMacTrackBar.Name = "redundancyMacTrackBar";
            this.redundancyMacTrackBar.Size = new System.Drawing.Size(252, 28);
            this.redundancyMacTrackBar.TabIndex = 6;
            this.redundancyMacTrackBar.TextTickStyle =
System.Windows.Forms.TickStyle.None;
            this.redundancyMacTrackBar.TickColor =
System.Drawing.Color.FromArgb(((int)(((byte)(148)))), ((int)(((byte)(146)))),
((int)(((byte)(148)))));
            this.redundancyMacTrackBar.TickHeight = 4;
            this.redundancyMacTrackBar.TickStyle =
System.Windows.Forms.TickStyle.None;
            this.redundancyMacTrackBar.TrackerColor =
System.Drawing.Color.FromArgb(((int)(((byte)(24)))), ((int)(((byte)(130)))),
((int)(((byte)(198)))));
            this.redundancyMacTrackBar.TrackerSize = new System.Drawing.Size(10,
16);
            this.redundancyMacTrackBar.TrackLineColor =
System.Drawing.Color.FromArgb(((int)(((byte)(90)))), ((int)(((byte)(93)))),
((int)(((byte)(90)))));
            this.redundancyMacTrackBar.TrackLineHeight = 3;
            this.redundancyMacTrackBar.Value = 19;
            this.redundancyMacTrackBar.ValueChanged += new
EConTech.Windows.MACUI.ValueChangedHandler(this.redundancyMacTrackBar_ValueChang
ed);
            this.redundancyMacTrackBar.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.redundancyMacTrackBar_MouseUp);
            //
            // allVolCountMacTrackBar
            //
            this.allVolCountMacTrackBar.BackColor =
System.Drawing.Color.Transparent;
            this.allVolCountMacTrackBar.BorderColor =
System.Drawing.SystemColors.ActiveBorder;
            this.allVolCountMacTrackBar.Font = new
System.Drawing.Font("Verdana", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.allVolCountMacTrackBar.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(123)))), ((int)(((byte)(125)))),
((int)(((byte)(123)))));
            this.allVolCountMacTrackBar.IndentHeight = 6;
            this.allVolCountMacTrackBar.Location = new System.Drawing.Point(6,
24);
            this.allVolCountMacTrackBar.Maximum = 15;
            this.allVolCountMacTrackBar.Minimum = 0;
            this.allVolCountMacTrackBar.Name = "allVolCountMacTrackBar";
            this.allVolCountMacTrackBar.Size = new System.Drawing.Size(252, 28);
            this.allVolCountMacTrackBar.TabIndex = 5;
            this.allVolCountMacTrackBar.TextTickStyle =
System.Windows.Forms.TickStyle.None;
            this.allVolCountMacTrackBar.TickColor =
System.Drawing.Color.FromArgb(((int)(((byte)(148)))), ((int)(((byte)(146)))),
((int)(((byte)(148)))));
            this.allVolCountMacTrackBar.TickHeight = 4;
            this.allVolCountMacTrackBar.TickStyle =
System.Windows.Forms.TickStyle.None;
            this.allVolCountMacTrackBar.TrackerColor =
System.Drawing.Color.FromArgb(((int)(((byte)(24)))), ((int)(((byte)(130)))),
((int)(((byte)(198)))));
            this.allVolCountMacTrackBar.TrackerSize = new
System.Drawing.Size(10, 16);
            this.allVolCountMacTrackBar.TrackLineColor =
System.Drawing.Color.FromArgb(((int)(((byte)(90)))), ((int)(((byte)(93)))),
((int)(((byte)(90)))));
            this.allVolCountMacTrackBar.TrackLineHeight = 3;
            this.allVolCountMacTrackBar.Value = 2;
```

```
            this.allVolCountMacTrackBar.ValueChanged += new
EConTech.Windows.MACUI.ValueChangedHandler(this.allVolCountMacTrackBar_ValueChan
ged);
            this.allVolCountMacTrackBar.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.allVolCountMacTrackBar_MouseUp);
            //
            // browser
            //
            this.browser.AutoValidate =
System.Windows.Forms.AutoValidate.EnablePreventFocusChange;
            this.browser.ListViewMode = System.Windows.Forms.View.List;
            this.browser.Location = new System.Drawing.Point(12, 131);
            this.browser.Name = "browser";
            treeNode1.Name = "";
            treeNode1.Text = "";
            treeNode2.ImageIndex = 18;
            treeNode2.Name = "backreg";
            treeNode2.SelectedImageIndex = 20;
            treeNode2.Text = "backreg";
            treeNode3.Name = "";
            treeNode3.Text = "";
            treeNode4.ImageIndex = 18;
            treeNode4.Name = "";
            treeNode4.SelectedImageIndex = 20;
            treeNode4.Text = "Code_Rid_Solomon_";
            treeNode5.Name = "";
            treeNode5.Text = "";
            treeNode6.ImageIndex = 18;
            treeNode6.Name = "Documents and Settings";
            treeNode6.SelectedImageIndex = 20;
            treeNode6.Text = "Documents and Settings";
            treeNode7.Name = "";
            treeNode7.Text = "";
            treeNode8.ImageIndex = 18;
            treeNode8.Name = "Завантаження";
            treeNode8.SelectedImageIndex = 20;
            treeNode8.Text = "Завантаження";
            treeNode9.Name = "";
            treeNode9.Text = "";
            treeNode10.ImageIndex = 18;
            treeNode10.Name = "Inprise";
            treeNode10.SelectedImageIndex = 20;
            treeNode10.Text = "Inprise";
            treeNode11.Name = "";
            treeNode11.Text = "";
            treeNode12.ImageIndex = 18;
            treeNode12.Name = "Program Files";
            treeNode12.SelectedImageIndex = 20;
            treeNode12.Text = "Program Files";
            treeNode13.ImageIndex = 33;
            treeNode13.Name = "Recycled";
            treeNode13.SelectedImageIndex = 34;
            treeNode13.Text = "Recycled";
            treeNode14.ImageIndex = 18;
            treeNode14.Name = "КОРЗИНА";
            treeNode14.SelectedImageIndex = 20;
            treeNode14.Text = "КОРЗИНА";
            treeNode15.ImageIndex = 18;
            treeNode15.Name = "Системна інформація";
            treeNode15.SelectedImageIndex = 20;
            treeNode15.Text = "Системна інформація";
            treeNode16.ImageIndex = 18;
            treeNode16.Name = "temp";
            treeNode16.SelectedImageIndex = 20;
            treeNode16.Text = "temp";
            treeNode17.Name = "";
            treeNode17.Text = "";
            treeNode18.ImageIndex = 18;
            treeNode18.Name = "WINDOWS";
```

```
            treeNode18.SelectedImageIndex = 20;
            treeNode18.Text = "WINDOWS";
            treeNode19.ImageIndex = 23;
            treeNode19.Name = "Локальный диск (C:)";
            treeNode19.SelectedImageIndex = 24;
            treeNode19.Text = "Локальный диск (C:)";
            this.browser.SelectedNode = treeNode19;
            this.browser.ShowFoldersButton = false;
            this.browser.ShowNavigationBar = false;
            this.browser.Size = new System.Drawing.Size(962, 432);
            this.browser.SplitterDistance = 398;
            this.browser.StartUpDirectoryOther = "C:\\";
            this.browser.TabIndex = 0;
            this.browser.Load += new System.EventHandler(this.browser_Load);
            //
            // testButton
            //
            this.testButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.testButton.Image =
global::RecoveryData.Properties.Resources.table_sql_view32451;
            this.testButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.testButton.Location = new System.Drawing.Point(111, 27);
            this.testButton.Name = "testButton";
            this.testButton.Size = new System.Drawing.Size(100, 97);
            this.testButton.TabIndex = 4;
            this.testButton.Text = "Перевірка цілісності";
            this.testButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.testButton.UseVisualStyleBackColor = true;
            this.testButton.Click += new
System.EventHandler(this.testButton_Click);
            //
            // repairButton
            //
            this.repairButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.repairButton.Image =
global::RecoveryData.Properties.Resources.medical_bag465471;
            this.repairButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.repairButton.Location = new System.Drawing.Point(210, 27);
            this.repairButton.Name = "repairButton";
            this.repairButton.Size = new System.Drawing.Size(100, 97);
            this.repairButton.TabIndex = 3;
            this.repairButton.Text = "Відновлення цілісності";
            this.repairButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.repairButton.UseVisualStyleBackColor = true;
            this.repairButton.Click += new
System.EventHandler(this.repairButton_Click);
            //
            // recoverButton
            //
            this.recoverButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.recoverButton.Image =
global::RecoveryData.Properties.Resources.redo6786987;
            this.recoverButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.recoverButton.Location = new System.Drawing.Point(309, 27);
            this.recoverButton.Name = "recoverButton";
            this.recoverButton.Size = new System.Drawing.Size(100, 97);
            this.recoverButton.TabIndex = 2;
            this.recoverButton.Text = "Читання даних з диску";
            this.recoverButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.recoverButton.UseVisualStyleBackColor = true;
            this.recoverButton.Click += new
System.EventHandler(this.recoverButton_Click);
            //
```

```
            // protectButton
            //
            this.protectButton.BackColor = System.Drawing.SystemColors.Control;
            this.protectButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.protectButton.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.protectButton.Image =
global::RecoveryData.Properties.Resources.Database_1_64x64e65768;
            this.protectButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.protectButton.Location = new System.Drawing.Point(12, 27);
            this.protectButton.Name = "protectButton";
            this.protectButton.Size = new System.Drawing.Size(100, 97);
            this.protectButton.TabIndex = 1;
            this.protectButton.Text = "Запис даних на диск";
            this.protectButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.protectButton.UseVisualStyleBackColor = false;
            this.protectButton.Click += new
System.EventHandler(this.protectButton_Click);
            //
            // ToolStripMenuItem
            //
            this.ToolStripMenuItem.Image =
global::RecoveryData.Properties.Resources.Exit;
            this.ToolStripMenuItem.Name = "ToolStripMenuItem";
            this.ToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
            this.ToolStripMenuItem.Text = "Вихід";
            this.ToolStripMenuItem.Click += new
System.EventHandler(this.выходToolStripMenuItem_Click);
            //
            // ToolStripMenuItem
            //
            this.тестБыстродействияToolStripMenuItem.Image =
global::RecoveryData.Properties.Resources.StartBenchmark;
            this.тестБыстродействияToolStripMenuItem.ImageScaling =
System.Windows.Forms.ToolStripItemImageScaling.None;
            this.ToolStripMenuItem.Name = "ToolStripMenuItem";
            this.ToolStripMenuItem.Size = new System.Drawing.Size(161, 22);
            this.ToolStripMenuItem.Text = "Тест швидкодії";
            this.ToolStripMenuItem.Click += new
System.EventHandler(this.тестБыстродействияToolStripMenuItem_Click);
            //
            // MainForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(986, 576);
            this.Controls.Add(this.coderConfigGroupBox);
            this.Controls.Add(this.testButton);
            this.Controls.Add(this.repairButton);
            this.Controls.Add(this.recoverButton);
            this.Controls.Add(this.protectButton);
            this.Controls.Add(this.browser);
            this.Controls.Add(this.menuStrip);
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.MainMenuStrip = this.menuStrip;
            this.MaximizeBox = false;
            this.Name = "MainForm";
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Підвищення надійності зберігання даних на носіях
інформації";
            this.Load += new System.EventHandler(this.MainForm_Load);
```

```
            this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.MainForm_FormClosing);
            this.menuStrip.ResumeLayout(false);
            this.menuStrip.PerformLayout();
            this.coderConfigGroupBox.ResumeLayout(false);
            this.redundancyGroupBox.ResumeLayout(false);
            this.redundancyGroupBox.PerformLayout();
            this.allVolCountGroupBox.ResumeLayout(false);
            this.allVolCountGroupBox.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.MenuStrip menuStrip;
        private System.Windows.Forms.ToolStripMenuItem ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem ToolStripMenuItem;
        private System.Windows.Forms.Button protectButton;
        private System.Windows.Forms.ToolStripMenuItem ToolStripMenuItem;
        private System.Windows.Forms.ToolStripSeparator
separatorToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem ToolStripMenuItem;
        private System.Windows.Forms.Button repairButton;
        private System.Windows.Forms.Button testButton;
        private System.Windows.Forms.GroupBox coderConfigGroupBox;
        private System.Windows.Forms.GroupBox redundancyGroupBox;
        private System.Windows.Forms.GroupBox allVolCountGroupBox;
        private EConTech.Windows.MACUI.MACTrackBar allVolCountMacTrackBar;
        private EConTech.Windows.MACUI.MACTrackBar redundancyMacTrackBar;
        private System.Windows.Forms.ToolTip toolTip;
        private System.Windows.Forms.ToolStripMenuItem ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem ToolStripMenuItem;
        internal FileBrowser.Browser browser;
        private System.Windows.Forms.ToolStripMenuItem ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem ToolStripMenuItem;
        private System.Windows.Forms.Button recoverButton;
    }
}
```

Файл Code_Rid_Solomon_Decoder.cs - декодер коду Ріда-Соломона

```
using System;
using System.Threading;

namespace Data_Center
{
    /// <summary>
    /// Клас декодера коду Ріда-Соломона
    /// </summary>
    public class Code_Rid_Solomon_Decoder : Code_Rid_Solomon_Base
    {
        #region Data

        // Масив булевських ознак "рядок матриці "FLog" тривіальна?"
        private bool[] FLogRowIsTrivial;

        // Список порядкових номерів наявних томів (нумерація з нуля)
        private int[] volList;

        #endregion Data

        #region Construction & Destruction

        /// <summary>
        /// Конструктор декодера за замовчуванням
        /// </summary>
        public Code_Rid_Solomon_Decoder()
        {
            // Створюємо об'єкт класу роботи з елементами поля Галуа
            this.eGF16 = new GF16();
        }

        /// <summary>
        /// Базовий конструктор декодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="volList">Список порядкових номерів наявних
томів</param>
        public Code_Rid_Solomon_Decoder(int dataCount, int eccCount, int[]
volList)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, volList,
(int)Code_Rid_Solomon_Type.Alternative);

            // Створюємо об'єкт класу роботи з елементами поля Галуа
            this.eGF16 = new GF16();
        }

        /// <summary>
        /// Розширений конструктор декодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="volList">Список порядкових номерів наявних
томів</param>
        /// <param name="codecType">Тип кодера коду Ріда-Соломона (по типу
матриці)</param>
        public Code_Rid_Solomon_Decoder(int dataCount, int eccCount, int[]
volList, int codecType)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, volList, codecType);
```

```
            // Створюємо об'єкт класу роботи з елементами поля Галуа
            this.eGF16 = new GF16();
        }

        #endregion Construction & Destruction

        #region Public Operations

        /// <summary>
        /// Установка конфігурації декодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="volList">Список порядкових номерів наявних
томів</param>
        /// <param name="codecType">Тип кодека кодера коду Ріда-Соломона (по
типу матриці)</param>
        /// <returns>Булевський прапор операції установки конфігурації</returns>
        public bool SetConfig(int dataCount, int eccCount, int[] volList, int
codecType)
        {
            int maxVolCount;

            // Установлюємо константи, що відповідають обраному режиму
            if (codecType == (int)Code_Rid_Solomon_Type.Dispersal)
            {
                maxVolCount = (int)Code_Rid_Solomon_Const.MaxVolCountDisp;

            } else
            {
                maxVolCount = (int)Code_Rid_Solomon_Const.MaxVolCountAlt;
            }

            // Перевіряємо конфігурацію на коректність
            if (
                    (dataCount > 0)
                 &&
                    (eccCount > 0)
                 &&
                    ((dataCount + eccCount) <= maxVolCount)
                 &&
                    (volList.Length >= dataCount)
                )
            {
                // Якщо основна конфігурація змінилася - сповіщаємо про це
                if (
                        (dataCount != this.n)
                     ||
                        (eccCount != this.m)
                     ||
                        (codecType != this.eCode_Rid_Solomon_Type)
                    )
                {
                    this.mainConfigChanged = true;
                }

                // Зберігаємо конфігурацію
                this.n = dataCount;
                this.m = eccCount;
                this.eCode_Rid_Solomon_Type = codecType;

                // Також перераховуємо кількість ітерацій всіх стадій підготовки
                double n = this.n;
                double m = this.m;

                // Нормалізуємо значення для розрахунку, щоб уникнути
переповнення змінних
                NormalizeNM(ref n, ref m);
```

```
                // Кількість ітерацій, що відслідковуються прогресом, на першій
стадії
                // залежить від типу використовуваної матриці
                if (this.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.Alternative)
                {
                    this.iterOfFirstStage = m;

                } else
                {
                    this.iterOfFirstStage = ((n * m) * n) + (n * ((n + m) + (n *
(n + m))));
                }

                this.iterOfSecondStage = (n * (((n - 1) * (n - 1)) + (n * n)));

                // Виділяємо пам'ять під масив булевських ознак "рядок матриці
"FLog" тривіальна?"
                this.FLogRowIsTrivial = new bool[dataCount];

                // Зберігаємо список наявних томів
                this.volList = volList;

                this.configIsOK = true;

            } else
            {
                //...указуємо на помилку конфігурації
                this.configIsOK = false;
            }

            return this.configIsOK;
        }

        /// <summary>
        /// Метод множення матриці кодування на вхідний прологарифмований вектор
        /// </summary>
        /// <param name="dataEccLog">Прологарифмований вхідний вектор (дані +
ecc)</param>
        /// <param name="data">Вихідний вектор (відновлені вихідні дані)</param>
        /// <returns>Булевський прапор результату операції</returns>
        public bool Process(int[] dataEccLog, ref int[] data)
        {
            // Якщо кодер зконфігуровано некоректно, обробка неможлива!
            if (!this.configIsOK)
            {
                return false;
            }

            // Копіюємо покажчик на масив експонент для скорочення часу обігу
            int[] GF16Exp = this.eGF16.GFExpTable;

            // Обчислення результату множення матриці на вектор
            for (int i = 0; i < this.n; i++)
            {
                // Якщо поточний рядок матриці не є тривіальної, робимо обробку
                if (!this.FLogRowIsTrivial[i])
                {
                    int mulSum = 0;     // Сума добутку рядка матриці на
стовпець
                    int i_n = i * this.n; // Зсув у масиві до елементів i-ой
рядка

                    for (int j = 0; j < this.n; j++)
                    {
                        mulSum ^= GF16Exp[this.FLog[i_n + j] + dataEccLog[j]];
                    }

                    data[i] = mulSum;
```

```
                } else
                {
                    data[i] = GF16Exp[dataEccLog[i]];
                }
            }

            return true;
        }

        #endregion Public Operations

        #region Private Operations

        /// <summary>
        /// Пошук матриці, зворотної до "FLog", методом Жорданових виключень
        /// (Дана модифікація методу може використовуватися тільки в тих
випадках,
        /// коли (-a) = (a), тому що через непотрібність пропущена стадія зміни
елементів),
        /// крім того, відсутній пошук ненульового розв'язного елемента (у
випадку
        /// роботи з матрицею Вандермонда наявність нуля на діагоналі - збій
кодека,
        /// тому ситуація з виявленням нуля сприймається винятково як помилка
        /// </summary>
        /// <returns>Булевський прапор результату операції</returns>
        private bool FInv()
        {
            // Обчислюємо розподіл відсотків ітерацій по стадіях для
            // коректної обробки відсотків
            double allStageIter   = this.iterOfFirstStage +
this.iterOfSecondStage;
            int percOfFirstStage  = (int)((100.0 * this.iterOfFirstStage)  /
allStageIter);
            int percOfSecondStage = (int)((100.0 * this.iterOfSecondStage) /
allStageIter);

            // Дана стадія повинна займати хоча б один відсоток
            // (для коректності розрахунків)
            if (percOfSecondStage == 0)
            {
                percOfSecondStage = 1;
            }

            // Обчислюємо значення модуля, що дозволить виводити відсоток
обробки
            // рівно при одиничному збільшенні для циклу по "k"
            int progressMod1 = this.n / percOfSecondStage;

            // Якщо модуль дорівнює нулю, то збільшуємо його до значення "1",
щоб
            // прогрес виводився на кожній ітерації
            if (progressMod1 == 0)
            {
                progressMod1 = 1;
            }

            // Цикл вибору розв'язного елемента "pivot"
            for (int k = 0; k < this.n; ++k)
            {
                // Якщо даний рядок тривіальна - просто переходимо на нову
ітерацію
                if (this.FLogRowIsTrivial[k])
                {
                    continue;
                }

                // Зсув у масиві до елементів k-ой рядка
```

```
                int k_n = k * this.n;

                // Індекс розв'язного елемента
                int pivotIdx = k_n + k;

                // Витягаємо розв'язний елемент
                int pivot = this.FLog[pivotIdx];

                // Якщо розв'язний елемент дорівнює нулю - матриця не має
звороткої
                if (pivot == 0)
                {
                    //...указуємо на помилку конфігурації
                    this.configIsOK = false;

                    // Активуємо індикатор актуального стану змінних-членів
                    this.finished = true;

                    // Установлюємо подію завершення обробки
                    this.finishedEvent[0].Set();

                    return false;
                }

                // Після добування розв'язного елемента поміщаємо на його місце
"1"
                this.FLog[pivotIdx] = 1;

                // Працюємо з рядками...
                for (int i = 0; i < this.n; i++)
                {
                    // Якщо перебуваємо на рядку розв'язного елемента -
переходимо
                    // на нову ітерацію
                    if (i == k)
                    {
                        continue;
                    }

                    // Зсув у масиві до елементів i-ой рядка
                    int i_n = i * this.n;

                    // Працюємо зі стовпцями...
                    for (int j = 0; j < this.n; j++)
                    {
                        // Якщо перебуваємо на стовпці розв'язного елемента -
переходимо
                        // на нову ітерацію...
                        if (j == k)
                        {
                            continue;
                        }

                        int idx = i_n + j;

                        //...а інакше робимо необхідні дії над матрицею:
                        // "A[i,j] = A[i,j] * pivot + A[i,k] * A[k,j]"
                        this.FLog[idx] = this.eGF16.Mul(this.FLog[idx], pivot) ^
this.eGF16.Mul(this.FLog[i_n + k], this.FLog[k_n + j]);
                    }
                }

                // Розподіл матриці на розв'язний елемент заміняємо множенням на
зворотний
                int pivotInv = this.eGF16.Inv(pivot);

                for (int i = 0; i < this.n; i++)
                {
                    // Зсув у масиві до елементів i-ой рядка
```

```
                    int i_n = (i * this.n);

                    for (int j = 0; j < this.n; j++)
                    {
                        int idx = i_n + j;

                        this.FLog[idx] = this.eGF16.Mul(this.FLog[idx],
pivotInv);
                    }
                }

                // Якщо є передплата на делегата відновлення прогресу -...
                if (
                        ((k % progressMod1) == 0)
                     &&
                        (OnUpdateCode_Rid_Solomon_MatrixFormingProgress != null)
                   )
                {
                    //...виводимо дані
                    OnUpdateCode_Rid_Solomon_MatrixFormingProgress((((double)(k
+ 1) / (double)this.n) * percOfSecondStage) + percOfFirstStage);
                }

                // У випадку, якщо потрібна постановка на паузу, подію
"executeEvent"
                // буде скинуто, і будемо на паузі аж до його появи
                ManualResetEvent.WaitAll(this.executeEvent);

                // Якщо зазначено, що потрібно вийти з потоку - виходимо
                if (ManualResetEvent.WaitAll(this.exitEvent, 0, false))
                {
                    // Указуємо, що декодер зконфігуровано некоректно
                    this.configIsOK = false;

                    // Активуємо індикатор актуального стану змінних-членів
                    this.finished = true;

                    // Установлюємо подію завершення обробки
                    this.finishedEvent[0].Set();

                    return false;
                }
            }

            return true;
        }

        /// <summary>
        /// Обчислення логарифмів значень інвертованої матриці
        /// <summary>
        private void LogFCalc()
        {
            // Працюємо з рядками...
            for (int i = 0; i < this.n; i++)
            {
                // Зсув у масиві до елементів і-ой рядка
                int i_n = i * this.n;

                // Працюємо зі стовпцями...
                for (int j = 0; j < this.n; j++)
                {
                    int idx = i_n + j;

                    this.FLog[idx] = this.eGF16.Log(this.FLog[idx]);
                }
            }
        }

        /// <summary>
```

```
        /// Заповнення матриці "FLog" (матриці декодера) даними
        /// </summary>
        protected override void FillFLog()
        {
            // Якщо довжина вектора наявних томів менше кількості,
            // необхідного для відновлення...
            if (this.volList.Length < this.n)
            {
                //...указуємо на помилку конфігурації
                this.configIsOK = false;

                // Активуємо індикатор актуального стану змінних-членів
                this.finished = true;

                // Установлюємо подію завершення обробки
                this.finishedEvent[0].Set();

                return;
            }

            // Виділяємо пам'ять під матрицю "FLog"
            this.FLog = new int[this.n * this.n];

            // Вектор лічильників всіх томів...
            int[] allVolCount = new int[this.n + this.m];

            //...і вектор ecc-томів для "затикання" пробілів, створених
            // загубленими основними томами
            int[] eccVolToFix = new int[this.m];

            // Лічильник кількості стертих основних томів
            int dataVolMissCount = this.n;

            // Ініціалізуємо масив лічильників всіх томів
            for (int i = 0; i < (this.n + this.m); i++)
            {
                allVolCount[i] = 0;
            }

            // Проводимо аналіз складу представлених томів на предмет наявності
основних
            for (int i = 0; i < this.n; i++)
            {
                // Обчислюємо номер поточного тому
                int currVol = Math.Abs(this.volList[i]);

                // Якщо номер тому відповідає припустимому діапазону
                if (currVol < (this.n + this.m))
                {
                    ++allVolCount[currVol];

                    // Якщо поточний том є основним, фіксуємо даний факт
                    if (currVol < this.n)
                    {
                        ---idataVolMissCount;
                    }

                } else
                {
                    // Указуємо на помилку конфігурації
                    this.configIsOK = false;

                    // Активуємо індикатор актуального стану змінних-членів
                    this.finished = true;

                    // Установлюємо подію завершення обробки
                    this.finishedEvent[0].Set();

                    return;
```

```
                }
            }

            // Перевіряємо лічильники томів на помилкове дублювання
            for (int i = 0; i < (this.n + this.m); i++)
            {
                // Якщо деякий том був зазначений більш ніж один раз...
                if (allVolCount[i] > 1)
                {
                    //...указуємо на помилку конфігурації
                    this.configIsOK = false;

                    // Активуємо індикатор актуального стану змінних-членів
                    this.finished = true;

                    // Установлюємо подію завершення обробки
                    this.finishedEvent[0].Set();

                    return;
                }
            }

            // Якщо перевірка на несуперечність не виявила проблем, починаємо
            // формувати матрицю "FLog"

            // Якщо основна конфігурація змінилася...
            if (this.mainConfigChanged)
            {
                if (this.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.Dispersal)
                {
                    //...робимо формування дисперсної матриці "D"
                    if (!MakeDispersalMatrix())
                    {
                        // Указуємо, що кодер зконфігуровано некоректно
                        this.configIsOK = false;

                        // Активуємо індикатор актуального стану змінних-членів
                        this.finished = true;

                        // Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();

                        return;
                    }

                } else
                {
                    //...робимо формування альтернативного заповнення матриці
"A"
                    if (!MakeAlternativeMatrix())
                    {
                        // Указуємо, що кодер зконфігуровано некоректно
                        this.configIsOK = false;

                        // Активуємо індикатор актуального стану змінних-членів
                        this.finished = true;

                        // Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();

                        return;
                    }
                }

                //...і скидаємо прапор
                this.mainConfigChanged = false;
            }
```

```
            // Для кожного загубленого основного тому шукаємо том для
відновлення
            for (int i = 0, j = 0; i < dataVolMissCount; i++)
            {
                // Рухаємося за списком томів доти, поки не знайдемо том для
                // відновлення для затикання "дірки" (основні томи мають номера
                // менше this.n (при нумерації з нуля!))
                while (this.volList[j] < this.n)
                {
                    j++;
                }

                // Зберігаємо номер тому для заміни загубленого основного тому
                eccVolToFix[i] = this.volList[j];

                j++; // j++ дозволяє перейти до наступного пошуку
            }

            // Працюємо по рядках матриці (в ідеалі, всі рядки повинні
заповнюватися
            // рядками з одиницею на головній діагоналі, що відповідає
відсутності
            // ушкоджень, але allVolCount укаже, якими є справи з наявністю
томів)
            for (int i = 0, e = 0; i < this.n; i++)
            {
                // Індекс рядка з дисперсної матриці, що буде поміщена в матрицю
кодування
                int DRowIdx;

                // Зсув у масиві до елементів i-ой рядка
                int i_n = i * this.n;

                // Якщо основний том відсутній, формуємо рядок матриці
Вандермонда
                if (allVolCount[i] == 0)
                {
                    // Обчислюємо номер рядка матриці Вандермонда, яку потрібно
вставити
                    // на місце даного рядка формованої матриці "FLog"
                    DRowIdx = eccVolToFix[e++];

                    // Указуємо, що даний рядок матриці "FLog" не тривіальна
                    this.FLogRowIsTrivial[i] = false;

                } else
                {
                    // Формуємо в матриці "FLog" нульовий рядок з одиницею на
головній діагоналі
                    // (відповідає наявному основному той)
                    DRowIdx = i;

                    // Указуємо, що даний рядок матриці "FLog" тривіальна
                    this.FLogRowIsTrivial[i] = true;
                }

                // Залежно від типу декодера беремо дані з відповідного масиву
                // (у ньому втримуються як рядки матриці Вандермонда, так і
"тривіальні" рядки,
                //  утримуючі нулі і єдиний елемент "1" на головній діагоналі)
                if (this.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.Dispersal)
                {
                    int bs = DRowIdx * this.n;

                    // Формування рядка в матриці кодування
                    // ("тривіальні" рядки вже втримуються в матриці "D", вони
вийшли
```

```
                    // "автоматично" на попередньому етапі обробки
MakeDispersal())
                    for (int j = 0; j < this.n; j++)
                    {
                        this.FLog[i_n + j] = this.D[bs + j];
                    }

                } else
                {
                    // Якщо це потрібно - формуємо "тривіальну" рядок...
                    if (this.FLogRowIsTrivial[i])
                    {
                        for (int j = 0; j < this.n; j++)
                        {
                            this.FLog[i_n + j] = 0;
                        }

                        this.FLog[i_n + i] = 1;

                    } else
                    {
                        int bs = (DRowIdx - this.n) * this.n;

                        //...а, інакше, беремо рядок матриці Вандермонда
                        for (int j = 0; j < this.n; j++)
                        {
                            this.FLog[i_n + j] = this.A[bs + j];
                        }
                    }
                }

                // У випадку, якщо потрібна постановка на паузу, подію
"executeEvent"
                // буде скинуто, і будемо на паузі аж до його появи
                ManualResetEvent.WaitAll(this.executeEvent);

                // Якщо зазначено, що потрібно вийти з потоку - виходимо
                if (ManualResetEvent.WaitAll(this.exitEvent, 0, false))
                {
                    //...указуємо на помилку конфігурації
                    this.configIsOK = false;

                    // Активуємо індикатор актуального стану змінних-членів
                    this.finished = true;

                    // Установлюємо подію завершення обробки
                    this.finishedEvent[0].Set();

                    return;
                }
            }

            // Знаходимо зворотну матрицю для "FLog"
            if (!FInv())
            {
                // Указуємо, що кодер зконфігуровано некоректно
                this.configIsOK = false;

                // Активуємо індикатор актуального стану змінних-членів
                this.finished = true;

                // Установлюємо подію завершення обробки
                this.finishedEvent[0].Set();

                return;
            }

            // Обчислюємо логарифми елементів інвертованої матриці
            LogFCalc();
```

```
            // Якщо є передплата на делегата завершення...
            if (OnCode_Rid_Solomon_MatrixFormingFinish != null)
            {
                //...повідомляємо, що екземпляр класу готовий до роботи
                OnCode_Rid_Solomon_MatrixFormingFinish();
            }

            // Активуємо індикатор актуального стану змінних-членів
            this.finished = true;

            // Установлюємо подію завершення обробки
            this.finishedEvent[0].Set();
        }

        #endregion Private Operations

        #region Public Properties

        /// <summary>
        /// Список порядкових номерів наявних томів
        /// </summary>
        public int[] VolList
        {
            get
            {
                if (!InProcessing)
                {
                    return this.volList;

                } else
                {
                    return null;
                }
            }
        }

        #endregion Public Properties
    }
}
```

Файл Code_Rid_Solomon_Encoder.cs - кодер коду Ріда-Соломона

```
using System;
using System.Threading;

namespace Data_Center
{
    /// <summary>
    /// Клас кодера коду Ріда-Соломона
    /// </summary>
    public class Code_Rid_Solomon_Encoder : Code_Rid_Solomon_Base
    {
        #region Construction & Destruction

        /// <summary>
        /// Конструктор кодера за замовчуванням
        /// </summary>
        public Code_Rid_Solomon_Encoder()
        {
            // Створюємо об'єкт класу роботи з елементами поля Галуа
            this.eGF16 = new GF16();
        }

        /// <summary>
        /// Базовий конструктор кодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        public Code_Rid_Solomon_Encoder(int dataCount, int eccCount)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount,
(int)Code_Rid_Solomon_Type.Alternative);

            // Створюємо об'єкт класу роботи з елементами поля Галуа
            this.eGF16 = new GF16();
        }

        /// <summary>
        /// Розширений конструктор кодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="codecType">Тип кодера коду Ріда-Соломона (по типу
матриці)</param>
        public Code_Rid_Solomon_Encoder(int dataCount, int eccCount, int
codecType)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, codecType);

            // Створюємо об'єкт класу роботи з елементами поля Галуа
            this.eGF16 = new GF16();
        }

        #endregion Construction & Destruction

        #region Public Operations

        /// <summary>
        /// Установка конфігурації кодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="codecType">Тип кодека кодера коду Ріда-Соломона (по
типу матриці)</param>
```

```
        /// <returns>Булевський прапор операції установки конфігурації</returns>
        public bool SetConfig(int dataCount, int eccCount, int codecType)
        {
            int maxVolCount;

            // Установлюємо константи, що відповідають обраному режиму
            if (codecType == (int)Code_Rid_Solomon_Type.Dispersal)
            {
                maxVolCount = (int)Code_Rid_Solomon_Const.MaxVolCountDisp;

            } else
            {
                maxVolCount = (int)Code_Rid_Solomon_Const.MaxVolCountAlt;
            }

            // Перевіряємо конфігурацію на коректність
            if (
                    (dataCount > 0)
                 &&
                    (eccCount > 0)
                 &&
                    ((dataCount + eccCount) <= maxVolCount)
                )
            {
                // Якщо основна конфігурація змінилася - сповіщаємо про це
                if (
                        (dataCount != this.n)
                     ||
                        (eccCount != this.m)
                     ||
                        (codecType != this.eCode_Rid_Solomon_Type)
                    )
                {
                    this.mainConfigChanged = true;
                }

                // Зберігаємо конфігурацію
                this.n = dataCount;
                this.m = eccCount;
                this.eCode_Rid_Solomon_Type = codecType;

                // Також перераховуємо кількість ітерацій всіх стадій підготовки
                double n = this.n;
                double m = this.m;

                // Нормалізуємо значення для розрахунку, щоб уникнути
переповнення змінних
                NormalizeNM(ref n, ref m);

                // Кількість ітерацій на першій стадії залежить від типу
використовуваної матриці
                if (this.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.Alternative)
                {
                    this.iterOfFirstStage = m;

                } else
                {
                    this.iterOfFirstStage = ((n * m) * n) + (n * ((n + m) + (n *
(n + m))));
                }

                this.iterOfSecondStage = 0; // У кодери немає інвертування
матриці

                this.configIsOK = true;

            } else
            {
```

```
                //...указуємо на помилку конфігурації
                this.configIsOK = false;
            }

            return this.configIsOK;
        }

        /// <summary>
        /// Метод множення матриці кодування на вхідний прологарифмований вектор
        /// </summary>
        /// <param name="dataLog">Прологарифмований вхідний вектор (вихідні
дані)</param>
        /// <param name="ecc">Вихідний вектор (надлишкові дані)</param>
        /// <returns>Булевський прапор результату операції</returns>
        public bool Process(int[] dataLog, ref int[] ecc)
        {
            // Якщо кодер зконфігуровано некоректно, обробка неможлива!
            if (!this.configIsOK)
            {
                return false;
            }

            // Копіюємо покажчик на масив експонент для скорочення часу обігу
            int[] GF16Exp = this.eGF16.GFExpTable;

            // Обчислення результату множення матриці на вектор
            for (int i = 0; i < this.m; i++)
            {
                int mulSum = 0;         // Сума добутку рядка матриці на
стовпець
                int i_n = i * this.n;   // Зсув у масиві до елементів i-ой рядка

                for (int j = 0; j < this.n; j++)
                {
                    mulSum ^= GF16Exp[this.FLog[i_n + j] + dataLog[j]];
                }

                ecc[i] = mulSum;
            }

            return true;
        }

        #endregion Public Operations

        #region Private Operations

        /// <summary>
        /// Заповнення матриці Вандермонда даними
        /// </summary>
        protected override void FillFLog()
        {
            // Якщо основна конфігурація змінилася...
            if (this.mainConfigChanged)
            {
                if (this.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.Dispersal)
                {
                    //...робимо формування дисперсної матриці "D"
                    if (!MakeDispersalMatrix())
                    {
                        // Указуємо, що кодер зконфігуровано некоректно
                        this.configIsOK = false;

                        // Активуємо індикатор актуального стану змінних-членів
                        this.finished = true;

                        // Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();
```

```
                        return;
                    }

                } else
                {
                    //...робимо формування альтернативного заповнення матриці
"A"
                    if (!MakeAlternativeMatrix())
                    {
                        // Указуємо, що кодер зконфігуровано некоректно
                        this.configIsOK = false;

                        // Активуємо індикатор актуального стану змінних-членів
                        this.finished = true;

                        // Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();

                        return;
                    }
                }

                // Виділяємо пам'ять під матрицю "FLog"
                this.FLog = new int[this.m * this.n];

                // Заповнюємо матрицю кодування
                for (int i = 0; i < this.m; i++)
                {
                    // Зсув у масиві до елементів i-ой рядка
                    int i_n = i * this.n;

                    // Залежно від типу кодера беремо дані з відповідного масиву
                    if (this.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.Dispersal)
                    {
                        // Формування рядка в матриці кодування
                        for (int j = 0; j < this.n; j++)
                        {
                            // У матрицю кодування поміщаємо логарифми її
вихідних елементів
                            // (для прискорення множення матриці на вектор)
                            this.FLog[i_n + j] = this.eGF16.Log(this.D[((this.n
+ i) * this.n) + j]);
                        }

                    } else
                    {
                        // Формування рядка в матриці кодування
                        for (int j = 0; j < this.n; j++)
                        {
                            int idx = i_n + j;

                            // У матрицю кодування поміщаємо логарифми її
вихідних елементів
                            // (для прискорення множення матриці на вектор)
                            this.FLog[idx] = this.eGF16.Log(this.A[idx]);
                        }
                    }

                    // У випадку, якщо потрібна постановка на паузу, подію
"executeEvent"
                    // буде скинуто, і будемо на паузі аж до його появи
                    ManualResetEvent.WaitAll(this.executeEvent);

                    // Якщо зазначено, що потрібно вийти з потоку - виходимо
                    if (ManualResetEvent.WaitAll(this.exitEvent, 0, false))
                    {
                        // Указуємо, що кодер зконфігуровано некоректно
```

```
                        this.configIsOK = false;

                        // Активуємо індикатор актуального стану змінних-членів
                        this.finished = true;

                        // Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();

                        return;
                    }
                }

                // Якщо є передплата на делегата завершення...
                if (OnCode_Rid_Solomon_MatrixFormingFinish != null)
                {
                    //...повідомляємо, що екземпляр класу готовий до роботи
                    OnCode_Rid_Solomon_MatrixFormingFinish();
                }

                //...і скидаємо прапор
                this.mainConfigChanged = false;
            }

            // Якщо є передплата на делегата завершення...
            if (OnCode_Rid_Solomon_MatrixFormingFinish != null)
            {
                //...повідомляємо, що екземпляр класу готовий до роботи
                OnCode_Rid_Solomon_MatrixFormingFinish();
            }

            // Активуємо індикатор актуального стану змінних-членів
            this.finished = true;

            // Установлюємо подію завершення обробки
            this.finishedEvent[0].Set();
        }

        #endregion Private Operations
    }
}
```

Файл About.cs - вікно довідки про програму

```
namespace RecoveryData
{
    partial class AboutForm
    {
        /// <summary>
        /// Необхідні змінні розробника.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true якщо керуючі ресурси повинні бути
розташовані, у іншому випадку  false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Необхідний метод для підтримки розробника - не модифікується
        /// зміст цього метода використовується редактором коду.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.developersListBoxdevelopersListBox = new
System.Windows.Forms.ListBox();
            this.Code_Rid_Solomon_IconTimer = new
System.Windows.Forms.Timer(this.components);
            this.okButtonXP = new PinkieControls.ButtonXP();
            this.SuspendLayout();
            //
            // developersListBoxdevelopersListBox
            //
            this.developersListBoxdevelopersListBox.BackColor =
System.Drawing.SystemColors.Control;
            this.developersListBoxdevelopersListBox.BorderStyle =
System.Windows.Forms.BorderStyle.None;
            this.developersListBoxdevelopersListBox.FormattingEnabled = true;
            this.developersListBoxdevelopersListBox.Items.AddRange(new object[]
{
            "БАКАЛАВРСЬКА РОБОТА",
            "",
            "На тему:",
            "",
            "Програмне забезпечення системи кібербезпеки мережевих дата-центрів
з перешкодостійким зберіганням інформації для забезпечення цілісності",
            "",
            "",
            "Керівник: Ликов Ілля Віталійович",
            "",
            "Розробив: студент Корнієць А.В.",
            "                  гр. КБ-20-3СК",
            "",
            "м. Кропивницький 2023"});
            this.developersListBoxdevelopersListBox.Location = new
System.Drawing.Point(11, 6);
            this.developersListBoxdevelopersListBox.Name =
"developersListBoxdevelopersListBox";
```

```
            this.developersListBoxdevelopersListBox.SelectionMode =
System.Windows.Forms.SelectionMode.None;
            this.developersListBoxdevelopersListBox.Size = new
System.Drawing.Size(330, 182);
            this.developersListBoxdevelopersListBox.TabIndex = 0;
            this.developersListBoxdevelopersListBox.TabStop = false;
            this.developersListBoxdevelopersListBox.SelectedIndexChanged += new
System.EventHandler(this.developersListBoxdevelopersListBox_SelectedIndexChanged
);
            //
            // Code_Rid_Solomon_IconTimer
            //
            this.Code_Rid_Solomon_IconTimer.Interval = 40;
            this.Code_Rid_Solomon_IconTimer.Tick += new
System.EventHandler(this.Code_Rid_Solomon_IconTimer_Tick);
            //
            // okButtonXP
            //
            this.okButtonXP.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(236)))),
((int)(((byte)(233)))), ((int)(((byte)(216)))));
            this.okButtonXP.DefaultScheme = true;
            this.okButtonXP.DialogResult =
System.Windows.Forms.DialogResult.None;
            this.okButtonXP.Hint = "";
            this.okButtonXP.Location = new System.Drawing.Point(266, 177);
            this.okButtonXP.Name = "okButtonXP";
            this.okButtonXP.Scheme = PinkieControls.ButtonXP.Schemes.Blue;
            this.okButtonXP.Size = new System.Drawing.Size(75, 23);
            this.okButtonXP.TabIndex = 0;
            this.okButtonXP.Text = "OK";
            this.okButtonXP.Click += new
System.EventHandler(this.okButtonXP_Click);
            //
            // AboutForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(350, 212);
            this.Controls.Add(this.okButtonXP);
            this.Controls.Add(this.developersListBoxdevelopersListBox);
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "AboutForm";
            this.ShowInTaskbar = false;
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "Про програму...";
            this.Load += new System.EventHandler(this.AboutForm_Load);
            this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.AboutForm_FormClosing);
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.ListBox developersListBoxdevelopersListBox;
        private System.Windows.Forms.Timer Code_Rid_Solomon_IconTimer;
        private PinkieControls.ButtonXP okButtonXP;

    }
}
```