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
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи Cloud-технології для NAS”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Сеїджанов А.
«____» _____ 2024 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 2024 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 *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

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

Сеїджанову Ахмеду
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи Cloud-технології для NAS*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 131-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи Cloud-технології для NAS*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2024 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	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Петренюк В.І.
(прізвище та ініціали)
Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Сеїджанов А.
(прізвище та ініціали)

АНОТАЦІЯ

Сеїджанов А. Програмне забезпечення системи Cloud-технології для NAS. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи Cloud-технології для NAS.

Метою розробки є програмне забезпечення системи Cloud-технології для NAS.

Результат роботи – програмна реалізація системи Cloud-технології для NAS.

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

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

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

Програму розроблено в середовищі Visual C#.

Ключові слова: комп'ютерна інженерія, Cloud-технології, NAS

ABSTRACT

Seidzhanov A. Cloud technology system software for NAS. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the Cloud technology system for NAS.

The purpose of the development is Cloud technology system software for NAS.

The result of the work is a software implementation of the Cloud technology system for NAS.

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 a PC with Windows 10/11 OS.

The program was developed in the Visual C# environment.

Keywords: computer engineering, Cloud technologies, NAS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП..... 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи..... 5
1.2 Область застосування..... 5
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти..... 7
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 24
2.3 Розгорнута постановка завдання 27
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 29
3.1 Опис функціонування системи 29
3.2 Розробка структурної схеми..... 31
3.3 Розробка функціональної схеми 40
3.4 Розробка діаграми процесів..... 44
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 46
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 46
4.2 Захист розробленого програмного забезпечення..... 60
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 65
6 ОСНОВНІ ВИСНОВКИ..... 70
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БЧХ	–	Боуз-Чоудхурі-Хоквінгхем
ЕОМ	–	електронно-обчислювальна машина
ЗП	–	запам’ятовуючий пристрій
КЗПІ	–	канал зберігання й передачі інформації
МНЗІ	–	методи підвищення надійності зберігання інформації
ПЕОМ	–	персональна електронно-обчислювальна машина
ПЗ	–	програмне забезпечення
СЗД	–	система зберігання даних
СНЗІ	–	система підвищення надійності зберігання інформації
ESRM	–	керування корпоративними ресурсами зберігання
SRM	–	керування ресурсами зберігання
SAN	–	адміністрування мереж

ВСТУП

Актуальність теми. Технології, покликані забезпечити максимально ефективне зберігання даних у системах зберігання даних (СЗД) середнього класу, поступово знаходять застосування в системах молодшого рівня, де все частіше використовуються дискові накопичувачі ємністю 6 Тбайт і реалізується підтримка SSD, установлюються 10-гігабітні мережні інтерфейси, передбачається інтеграція із хмарними сервісами зберігання, розширюються функціональні можливості при одночасному спрощенні керування, що стає більше зручним.

Зберігання даних – найважливіше завдання, рішення якого повинна забезпечувати IT-інфраструктура практично будь-якої компанії, включаючи підприємства малого й середнього бізнесу (SMB). За прогнозом аналітичної компанії TechNavio, у поточному й наступному році середньорічні темпи росту світового ринку мережних систем зберігання (Network Attached System, NAS) у грошовому вираженні будуть перевищувати 50%, чому повинне сприяти впровадження спеціалізованих серверів зберігання даних. Відповідно до дослідження IDC EMEA Quarterly Disk Storage Systems Tracker, в I кварталі 2016 року загальна ємність зовнішніх дискових систем зберігання даних, поставлених на український ринок, склала більше 83,1 Пбайт, а їхня сукупна вартість перевищила 117,76 млн доларів. У цілому в порівнянні з аналогічним періодом минулого року український ринок СЗД продемонстрував помірний ріст (3,5%) у грошовому вираженні й істотний (35,3%) у гігабайтах. При цьому середня вартість розраховуючи на гігабайт продовжує знижуватися завдяки еволюційному розвитку технологій зберігання.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи Cloud-технології для NAS.

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

– Огляд існуючих систем Cloud-технології для NAS.

– Дослідження системи Cloud-технології для NAS.

– Програмна реалізація системи Cloud-технології для NAS.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі Cloud-технології для NAS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи Cloud-технології для NAS, 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

В останні кілька місяців новинки в області мережних систем зберігання молодшого класу з'явилися практично скрізь, де застосовуються NAS для малого й середнього бізнесу – від класичних мережних сховищ і файлових серверів до систем резервного копіювання й відеореєстраторів. Однак перспективи поки не радують. Ситуація на українському ринку СЗД для SMB не відрізняється від інших сегментів. Її можна визначити одним словом – спад. У зв'язку зі скороченням обсягів бізнесу інвестиції в IT-рішення втрачають актуальність. Цього року спостерігається зниження попиту в грошовому вираженні й, як наслідок, його зсув убік дешевих рішень. На ринку SMB ця тенденція проявляється особливо чітко, оскільки невеликі підприємства більш оперативно реагують на зміну умов ведення бізнесу». У такій ситуації організаціям доводиться вирішувати непрості проблеми зберігання постійно зростаючого (на відміну від бюджетів) обсягу структурованих і неструктурованих даних. Причому сьогодні це пропонується робити централізовано, на мережних сховищах, що спрощує резервне копіювання й пошук інформації й дозволяє застосовувати єдині правила відносно способів і об'єктів зберігання. А реплікація даних NAS між віддаленими офісами або автоматичною передачею важливих файлів у центральний офіс або в хмарне сховище забезпечує захист від відмов.

1.2 Область застосування

Щоб мінімізувати виникаючі при хмарному зберіганні ризики, компанії можуть зберігати коштовні дані у своєї IT-інфраструктурі, а менш важливі – наприклад, архівні дані або резервні копії – відправляти в хмару (Cloud Backup).

Відповідно до досліджень, у будь-якій мережі активно використовується близько 20-30% даних: їх прийнято називати «гарячими». Інші 70-80% практично не змінюються й не використовуються («холодні» дані), а виходить, має сенс зберігати їх в архівних системах або в хмарі. Тому NAS всі частіше інтегруються із хмарними сервісами.

Хмарні сервіси вже затребувані. Їхнє використання дозволяє не тільки швидко одержувати потрібну ємність зберігання даних, але й при необхідності скорочувати її, оперативно реагуючи на зміни умов ведення бізнесу. Крім того, простіше стає надати доступ до даних для клієнтів і партнерів. Повсюдне поширення таких сервісів стримується недостатньою пропозицією з боку українських провайдерів і упередженням проти використання хмарних технологій, які найчастіше вважаються менш надійними. Не сприяють росту довіри й випадки витоку конфіденційних даних із хмарних сервісів. Провайдерам ще має бути переконати користувачів у надійності й, саме головне, безпеки надаваних ними сервісів.

Однак затребуваність послуг зберігання даних у режимі онлайн буде рости в міру розвитку інших хмарних сервісів. У невеликих компаніях сервіси онлайнового зберігання інформації можуть застосовуватися замість власних систем зберігання, у середні вони, швидше за все, стануть додатковими сховищами. Найбільш імовірні напрямки інтеграції малих СЗД із хмарними сервісами – резервне копіювання в хмару й синхронізація окремих папок із хмарним зберіганням з метою простого надання доступу до даних.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи Cloud-технології для NAS, 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

Домашній користувач рідко зіштовхується з мережним сховищем (Network Attached Storage) у побуті, тому часто вважає його установку у своїй квартирі зайвою, нерозумною й незрозумілою примхою. Тим часом, існує маса сценаріїв роботи з домашньою електронікою й побутовою технікою, використання NAS у які різко оптимізує процес і забезпечує йому інший рівень безпеки.

Які явні ознаки того, що використання мережного сховища у вашім будинку обернеться благом? Цих ознак небагато, тому перелічимо їх усе.

По-перше, і це – головне: NAS може виявитися корисний там, де живуть люди, що використовують відразу кілька комп'ютерів, об'єднаних у локальну мережу й систему «розумний будинок». Говорячи про «декілька комп'ютерів», важливо відзначити, що зараз під це визначення підходить майже кожна родина, де використовують смартфони, планшетні комп'ютери, ноутбуки, медіаплеєри, ігрові приставки й телевізори з функцією Smart TV.

По-друге, NAS значно спрощує життя людям, захопленим кінематографом і які збирають кінотеку фільмів у максимально доступною зараз якістю. Сюди ж можна віднести власників домашніх кінозалів, обладнаних більшими 3D-телевізорами, 3D-проекторами й багатоканальними акустичними системами.

По-третє, наявність у мережного сховища будинку бідують професіонали, що працюють приватним порядком з відео, фотографією й звуком: оператори, монтажери, режисери, фотографи, композитори, ді-джеї, продюсери з домашніми студіями звукозапису...

По-четверте, NAS готовий стати неодмінним атрибутом інтер'єра будинку там, де хазяї хочуть тримати свій власний torrent-сервер, хостить сайт, допомагати сусідам по локальної будинкової мережі з організацією поштового сервера.

Нарешті, в-п'ятих, NAS добре себе покаже в житло з великою площею й великою прибудинковою територією – приймаючи відеопотоки від камер спостереження.

Коротенько позначивши перспективні напрямки застосування мережних сховищ для сценаріїв домашнього використання, перейдемо до подробиць...

NAS у ролі домашнього медіа-сервера

Група сценаріїв домашнього застосування мережного сховища для організації доступу до даних так чи інакше пов'язана з поняттям «сервера». Сучасні NAS відмінно підходять для організації на їхній базі серверів всілякої спрямованості – як використовуваних у рамках локальної домашньої мережі, так і зв'язку, що мають канал, з інтернетом.

Саме очевидний і самий популярний сценарій цієї групи – застосування мережного сховища як медіа-сервера. Цей сценарій припускає каталогізоване складування на NAS відео- звукових і графічних файлів для доступу до них комп'ютерів, телевізорів, мережних медіацентрів, ноутбуків, планшетів і смартфонів, що перебувають у будинку. Фактично, це – розважальна грань роботи так званого «розумного будинку»: мережної інфраструктури, що зв'язує всі домашні гаджети в єдину функціональну систему.

Використовуючи NAS як медіа-сервер, буває корисно довідатися: з якими конкретно сучасними мобільними пристроями він сумісний, є в його програмній платформі убудовані механізми каталогизування контенту, як здійснюється доступ і на якій швидкості, є чи можливість установити в налаштуваннях сховища доступ з додатка iTunes (iTunes-сервер)?

Наступний по популярності сценарій домашнього використання NAS – організація так званого torrent-сервера. Функції torrent-сервера припускають, що

сховище буде працювати в режимі цілодобового підключення до інтернету для завантаження нових файлів через пірінгову мережа й забезпечення доступу до уже завантаженого контенту з боку інших користувачів. Робота torrent-сервера на домашньому NAS жадає від програмної платформи мережного сховища наявність відповідних механізмів. Як мінімум убудованих утиліт, що дублюють функції програм torrent-клієнтів.

NAS у ролі спостережного пункту

Нарешті, застосовувати NAS у зв'язуванні з камерами спостереження будинку можуть ті люди, чиє житло має площа в кілька сотень квадратних метрів і двір, вимірюваний гектарами.

Цей сценарій домашнього використання не можна назвати занадто розповсюдженим, тому коротенько розповімо про деякі його нюанси.

Почнемо з того, що, вибираючи NAS для організації сервера відеоспостереження за допомогою вивчення його характеристик, ви можете бути здивовані тому, що в налаштуваннях цього пункту не виявиться. Як тільки що було сказано, організація сервера спостереження – не сама затребувана функція мережного сховища. Тому деякі виробники NAS роблять її програмно факультативної: щоб одержати бажану функціональність, користувачеві потрібно скачати й установити утиліту-розширення.

Ще один нюанс стосується безпосередньої роботи мережного сховища з камерами спостереження. Звичайно, не варто згадувати, що мова йде про IP-Камери, які відрізняються від звичайних тем, що використовують бездротове з'єднання з повноцінним закріпленням за кожним пристроєм IP-Адреси й кодують відеопотік з оптимальним співвідношенням якості й обсягу даних.

Робота NAS як сервер відеоспостереження дуже розрізняється від моделі до моделі й від виробника до виробника. Основний параметр отут – кількість виділених каналів. Цей параметр може бути як постійним, так і розширюваним. В останньому випадку виробник організовує процес через продаж ліцензійних кодів, що відкривають можливість приєднувати до сервера нові камери.

Рисунок 2.1 – NAS у ролі спостережного пункту

NAS у ролі аудіо- і відеоархіву

На відміну від будь-якої іншої периферійної техніки для комп'ютера, що використовує лише функціонально необхідні електронні компоненти, мережне сховище являє собою спеціалізований, але самий справжній комп'ютер. Тому очевидно, що робота на домашньому ПК у парі з NAS автоматично припускає створення в будинку локальної мережі.

Експлуатація домашньої мережі жадає від користувача нехай мінімальних, але конкретних знань про її адміністрування. Обзаводиться таким досвідом заради підключення NAS з одному єдиному комп'ютеру у квартирі, напевно, не має сенсу – тому покупка NAS і його інтеграція ефективна там, де вже

налаштована локальна домашня мережа з декількома пристроями й маршрутизатором.

При всіх сценаріях використання мережного сховища будинку від пристрою потрібно справно виконувати дві основні функції – зберігати більші обсяги інформації з максимально можливим рівні безпеки й забезпечувати доступ до неї за схемою, певної адміністратором.

По черзі почнемо ми зі сценаріїв, де основна функція – це зберігання.

Зберігання відеофайлів в абсолютній більшості випадків домашнього використання NAS означає створення персональної фільмотеки. У сучасних умовах, коли тарифи на безлімітний доступ в інтернет досить невеликі, а швидкість з'єднання дозволяє дивитися кіно в потоковому режимі, колекціонування відеофайлів на локальному жорсткому диску може здатися архаїзмом. Однак два моменти змушують подивитися на питання під іншим кутом.

Один з них стосується якості доступу в інтернет. Не скрізь у нашій країні він однаковий – особливо отут важко доводиться сільським жителям і жителям невеликих міст. Який зв'язок між сільськими жителями й NAS? Цілком пряма: зараз навколо багатьох обласних і районних центрів з'являються котеджні селища й селища, що складаються з таунхаусів. Їхні мешканці здебільшого цілком у стані мати не просто комп'ютер, а цілий парк електронних пристроїв. Однак доступ в інтернет для всіх цих пристроїв – не дивлячись на їхню кількість і назви брендів – залишається на рівні сільських стандартів. От і виходить, що використання NAS для зберігання фільмотеки – це абсолютно життєвий сценарій для жителів котеджів, що перебувають у сільській місцевості, приватних будинків і таунхаусів.

Другий момент створення фільмотеки на базі мережного сховища стосується обсягу видеофайлів і особливим вимогам до швидкості доступу до них. Як уже було сказано вище, у великих містах максимальна швидкість широкополосного доступу в інтернет дозволяє дивитися фільми потоком у досить

непоганій якості. Однак якість і вимоги до швидкості доступу постійно ростуть. Стандарт Full HD з п’ятиканальним звуком уже став щоденністю. У штатному порядку виходять 3D-фільми. Продажу телевізорів Ultra HD набирають оберти – а з ними з'являється можливість випускати кінострічки з дозволом 3840 на 2160 крапок. Захоплений киноман сьогодні просто змушений збирати свою колекцію на базі мережного сховища, якщо в його пріоритетах значиться якість зображення – а значить розмір файлу і його бітрейт при відтворенні.

Завершуючи тему швидкості доступу, висловимо пари рекомендацій, що стосуються характеристик різних моделей NAS. Сучасні мережні сховища для інтеграції в мережу використовують або звичайне провідне з'єднання через порт Ethernet, або його в парі з бездротовим модулем Wi-Fi. Останній, безумовно, зручний – особливо для мобільних пристроїв, але навіть при сумісності зі стандартом 802.11n не може забезпечити швидкість передачі даних вище 600 мегабіт у секунду. Провідне ж з'єднання вкупі з гігабітним мережним адаптером дозволяє впевнено себе почувати навіть при перегляді з жорстких дисків NAS фільмів і передач у дозволі Ultra HD і форматі 3D.

Відео, записане й, що зберігається на дисках мережного сховища, прерогатива не одних лише кіноаматорів. Ще один сценарій використання NAS будинку – це створення архіву при роботі приватним відеооператором, монтажером, режисером…Не секрет, що виробництво фільмів передбачає нагромадження великої кількості відеоматеріалу, обсяг якого продовжує рости – завдяки впровадженню в життя камер, що знімають в Ultra HD або «сиром» форматі RAW.

Використання мережного сховища як домашній архів робочих матеріалів може бути вкрай зручно для людей, індивідуальним порядком які займаються також фотографією або музикою.

Спочатку поговоримо про фотографів. Ні для кого не секрет, що робота професійного фотографа припускає надмірність матеріалу. Приміром, американські майстри, що спеціалізуються на зйомці весіль, у ході одного

замовлення роблять до 5000 кадрів. Це – колосальний обсяг інформації – з урахуванням використання сучасних апаратів із багатопіксельними матрицями й використанні «сирого» RAW. Вітчизняні весілля мають свою специфіку – на них фотограф звичайно відпрацьовує цілий день. У результаті вихід по кадрам у нього не занадто відрізняється від показників американських колег.

Великий обсяг даних у вигляді тисяч кадрів генерують і інші фото-специ: працюючі в жанрі репортажу, зайняті в зйомці портретів, студійній і предметній зйомці, фотографи-мандрівники.

При постійному режимі зайнятості обсяг архіву стає таким, що тримати, каталогізувати й обробляти його на комп'ютері стає просто незручно. І установка мережного сховища рятує ситуацію.

Ті ж слова можна повною мірою віднести до людей, що займається музикою. Це – композитори, що працюють індивідуально ді-джеї, люди, що мають вдома студію звукозапису. Їхня необхідність мати більшу фонотеку, містити її в каталожному порядку, зберігати безліч версій власних (або чужих) виробничих записів найкраще забезпечується мережним сховищем.

При постійному збільшенні збереженої на домашньому NAS інформації користувачеві потрібно звернути увагу на кількість настановних місць для жорстких дисків. Звичайно, більшість підсвідомо припускає, що мережному сховищу для будинку навряд чи потрібна можливість підключення більше одного або двох дисків. Однак, як нам здається, завжди краще просто купити новий HDD і вставити його в шахту, чим думати над придбанням диска помітно більшого, ніж зараз, обсягу й вишукувати спосіб копіювання на нього вже наявної інформації. Нарешті, нагадаємо, що NAS із чотирма й більше «посадковими місцями» може повною мірою використовувати всі переваги системи злиття дискового обсягу в масив RAID. Втім, про це – небагато пізніше.

NAS у ролі сховища резервних копій

Ще один, досить важливий по своїй суті, сценарій домашнього використання мережного сховища актуальний там, де в досить великої родини в

арсеналі виявляється багато сучасної техніки. Смартфони, планшетні комп'ютери, ноутбуки, настільні комп'ютери – всі вони відмінно допомагають у роботі й розвазі…поки щось усередині піде не так. Іноді із проблемою вдається впорається легко, іноді – доводиться переустановлювати систему й проходити довгий і болісний шлях відновлення програмного середовища.

Максимально спростити життя в цій критичній ситуації здорово допомагає мережне сховище із записами образів програмних середовищ всіх наявних у будинку комп'ютерів і гаджетів. Сучасні електронні пристрої й операційні системи здебільшого дозволяють одержати файли, що містять точний образ програмного середовища, що включає всі налаштування, всі програми, всі актуальні документи й так далі. Повернути звичне положення справ після системної відмови особистого електронного помічника – елементарно.

Треба відзначити, що програмна платформа якісних NAS досить серйозно підходить до питанню резервного копіювання. Адміністративні налаштування сховищ дозволяють настроїти регулярне відновлення архівних образів – як у ручному, так і в автоматичному режимі.

Ще більше убезпечити свій спокій допомагає застосування для дисків, установлених у домашньому NAS, режиму роботи як масив RAID. Система RAID має наступні можливості: об'єднання простору всіх дисків із кратним збільшенням доступу до нього (RAID 0), повне дзеркалювання (повне копіювання) пар дисків (RAID 1), об'єднання простору з виділенням одного з дисків під зберігання контрольних сум (RAID 5, доступно для мережних сховищ із чотирма й більше слотами установки HDD).

NAS у ролі поштової відділення, web-хостингу

Менш розповсюджені, але все-таки актуальні для будинку сценарії – це створення на їхній базі web-сервера для запуску у всесвітній мережі особистого сайту (одиночного блога, онлайнового магазина, стартапа…),поштового сервера або сервера обслуговування камер відеоспостереження. Організовувати в домашніх умовах поштовий сервер може бути корисно, якщо жителі

багатоквартирного будинку організували свою власну локальну мережу. Що стосується web-сервера вдома, те отут, як нам здається, ми підходимо до грані, що відокремлює сценарії приватного й корпоративного використання NAS. У зв'язку із цим, користувачеві рекомендується звернути увагу на обчислювальну потужність обираного для організації web-сервера мережного сховища. У першу чергу вона залежить від технічної платформи й архітектури процесора.

Представлені на ринку сучасні NAS працюють під керуванням центральних процесорів двох архітектур: ARM і x86. Остання представлена практично винятково процесорами Intel Atom. Саме вони кращі в сценаріях, які припускають велику кількість підключень до сервера з інтенсивним обміном інформацією. Крім того, NAS на платформі x86 мають більше звичну для користувачів персональних комп'ютерів конструкцію й більше придатні для ремонту й модернізації в домашніх умовах.

Переваги мережних сховищ на платформі ARM – це вартість, мінімальне тепловиділення (а значить – проста й безшумна система охолодження), практично нульове споживання електроенергії в стані спокою системи й дуже скромне – у стані роботи.

Buffalo LinkStation LS410D

Модель LS410D є молодшою із серії LinkStation 400, що представляє найбільш продуктивні накопичувачі для домашнього застосування й малого бізнесу. Виробник на своєму сайті обіцяє «утроє більшу швидкість у порівнянні зі стандартним NAS», уточнюючи цю заяву більше конкретною цифрою: «до 100 Мбіт/с» (напис на впакуванні трохи скромніше: «до 80 Мбіт/с»). Роботою накопичувача управляє LinkStation NAS System, недавно розробленою компанією Buffalo. Крім багатих можливостей керування даними, до яких ставиться підтримка різних протоколів доступу, системи контролю доступу Active Directory, Apple Time Machine і прямого копіювання з USB-накопичувачів, вона працює під керуванням тої ж операційної системи, що й всі інші моделі компанії. Завдяки цьому ми одержуємо єдині принципи роботи й уніфікований інтерфейс.

У ногу з модерновими тенденціями пропонується підтримка різноманітних «хмарних» сервісів, таких як синхронізація з Flickr, або створення власного «хмари» з доступом з довільної крапки всесвітньої Мережі. В автоматичному режимі виробляється завантаження файлів убудованим клієнтом BitTorrent, управляти яким можна за допомогою додатка для Android або iOS. Є також мультимедійний DLNA-сервер і сервер iTunes. Для мережного диска початкового рівня, такий богатий набір можливостей виглядає досить привабливо. А як щодо апаратних характеристик?

Конструкція LS410D досить компактна, корпус ненабагато більше габаритів тридюймового вінчестера, установленого усередині. Всі варіанти моделі заздалегідь укомплектовані дисками, серед яких є варіанти 2, 3, або 4 Тбайт. У наше розпорядження потрапив перший з перерахованих варіантів, із установленим усередині двухтерабайтником Seagate Barracuda 7200.14. На «зализаних» гранях передньої частини корпуси проглядаються дві білі смужки-індикатора, що подають світлові сигнали про стан пристрою, і червона смуга на самому верху, що є просто елементом дизайну. Всі інші органи керування згрупувалися позаду, включаючи компактний 40-міліметровий вентилятор охолодження й вимикач живлення. Незважаючи на гадану «аппаратність», вимикач не має зв'язку із проводами живлення, а всього лише подає сигнал на контролер диска, тобто працює програмно, тому завершення роботи при ручному відключенні виконується коректно.

Із зовнішніх інтерфейсів є тільки гігабітний Ethernet і один USB-порт ревізії 2.0, що підтримує підключення додаткового диска або принтера. Живлення здійснюється від зовнішнього адаптера невеликих розмірів. Рознімання для підключення адаптера постачений спеціальним гачком, за який можна зачепити проведення для запобігання випадкового висмикування. Є також слот для замка Kensington, хоча складно представити ситуацію, у якій може придатися кріплення пристрою металевим тросом.

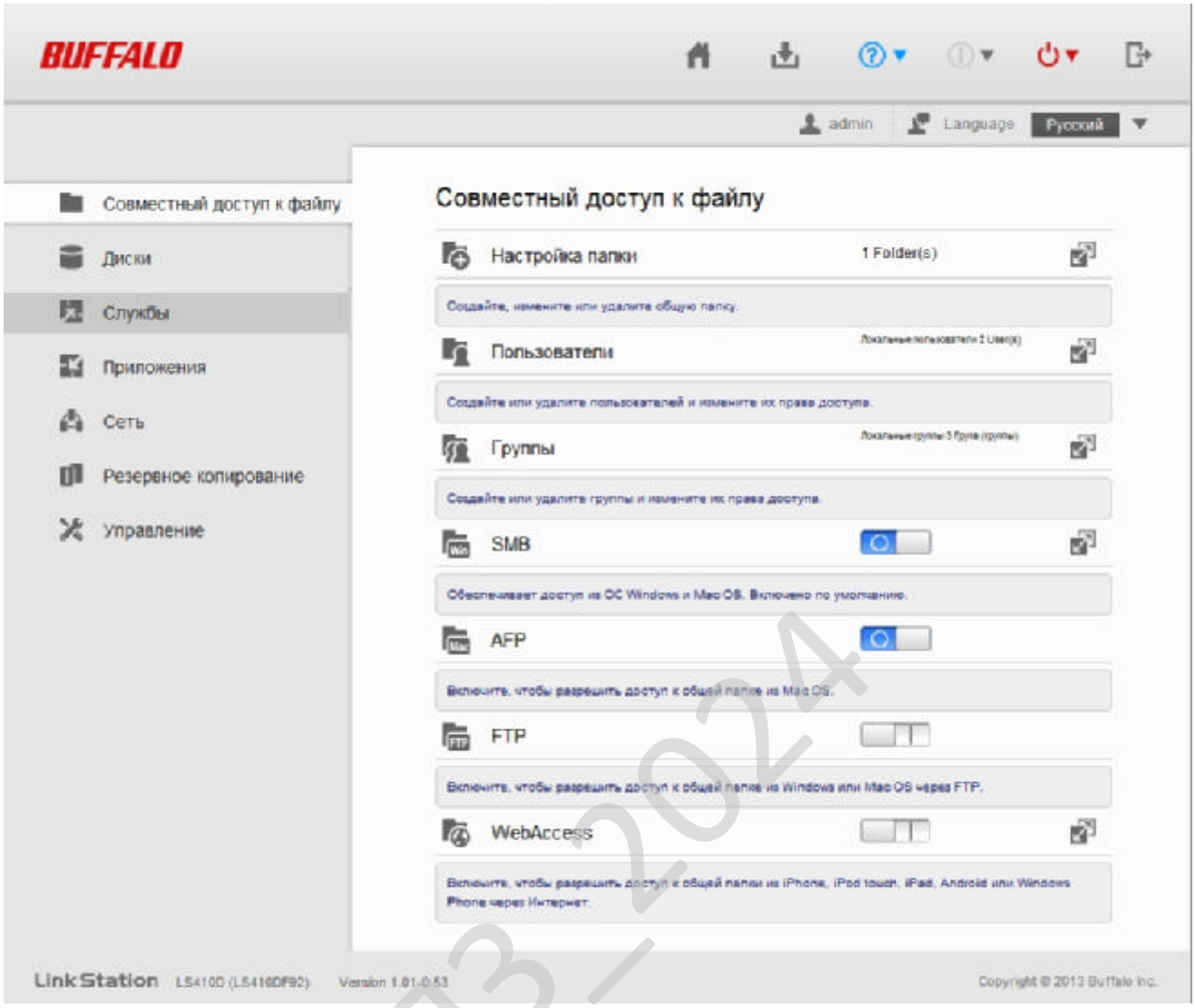

Рисунок 2.2 – Інтерфейс системи налаштування Buffalo LinkStation LS410D

Розташувати пристрій на столі можна як горизонтально, так і вертикально: у першому випадку на мікроскопічних гумових ніжках, у другому – на тонкій гумовій прокладці, що займає всю бічну грань, за винятком великого шильдика з назвою моделі й іншою технічною інформацією. У горизонтальній орієнтації шильдик виявляється збоку й добре помітний, але зовсім не прикрашає зовнішній вигляд. Маленький вентилятор працює досить галасливо, незважаючи на «інтелектуальне» керування швидкістю. Також багато шуму створює обертання шпинделя диска, позбавити від якого тонкі прокладки не мають сил. Будучи

встановленим на стіл, накопичувач змушує його резонувати низькочастотним гулом. Якщо підкласти під нього м'який гумовий коврик, проблема зникає.

Веб-інтерфейс убудованої ОС виконаний по модульному принципі: основний екран складається із квадратних плиток, кожна з яких управляє однієї із заданих функцій, наприклад, DLNA-сервером, Torrent-клієнтом або пристроями USB. Кількість і розташування плиток можна довільно задавати в налаштуваннях. При заході усередину якого-небудь розділу ми бачимо ліворуч основне меню, а в іншій частині екрана – детальну інформацію з обраного пункту. Є контекстна довідка: наприклад, над кнопкою «Завантажити» спливає пояснення: «Завантажити». Якщо потрібно змінити параметри, це робиться в спливаючому вікні, що з'являється при натисканні на обраний пункт. Для роботи інтерфейс вимагає останніх версій браузерів IE, Firefox або Google Chrome, як повідомляє нас спливаюче вікно, але насправді все добре й у старенької Opera 12. Промальовування відбувається досить швидко, а от відгук від накопичувача приходять із помітною затримкою: наприклад, така проста операція, як завдання пароля, займає кілька секунд. При цьому відображається спливаюче вікно із пропозицією почекати. Це трохи спантеличує, тому що швидкодії встановленого процесора, 1, 2-гігагерцового Marvell Armada 370, повинне із запасом вистачати для подібних операцій.

При тестуванні програмою Intel NAS Performance Toolkit накопичувач продемонстрував цілком гідні показники. Правда, результати сильно залежать від налаштувань розміру фрейму мережі, як у мережному адаптері ПК, так і в самому накопичувачі – вони повинні бути якнайбільше, але при цьому відповідати один одному. Фактично, він став лідером нашого огляду, трохи обійшовши пристрій Synology, побудований на тім же самому процесорі. Швидкість запису навіть перевершила заявлені 100 Мбіт/с (по іронії долі по цьому параметрі він злегка програє DS114), швидкість читання ж трохи скромніше: близько 60 Мбіт/с для одного потоку або файлу, а з ростом числа потоків сумарна швидкість теж трохи росте, досягаючи при чотирьох потоках 77 Мбіт/с. Цей факт, як і незвично висока

швидкість у режимі фотоальбому (10 Мбіт/с), говорять про гарну масштабованість пропускної здатності під інтенсивним навантаженням. З обліком досить демократичної ціни, LinkStation LS410D може бути вдалим вибором.

Synology DS114

Мережне сховище DS114 від Synology – теж однодискова модель, ліліпут серед корпоративних монстрів на 12 і більше дисків, що випускаються компанією. Як і більше місткі побратими, працює вона під керуванням однієї й тої ж дискової ОС – DiskStation Manager версії 5.0. Правда, на відміну від виробу Buffalo, що поставляється в готовому виді, тут перед початком роботи потрібно скачати й установити цю ОС. Очевидно, що зв'язано це з відсутністю диска, тому що ПЗ встановлюється безпосередньо на нього, а докуповувати його потрібно самостійно. Ми використовували для наших експериментів вінчестер WD Red обсягом 4 Тбайт – типовий кандидат для домашнього NAS з невеликим енергоспоживанням і нагріванням, і швидкістю обертання шпинделя 5900 про/хв, замаскованої виробником мрячним терміном «IntelliPower».

Установка диска не займає багато часу, хоча й вимагає наявності хрестоподібної викрутки. Корпус DS114 набагато більше об'ємний, всі деталі в ньому розташовані вільно, що поліпшує охолодження. Назва компанії на бічній грані й декоративній сходинці на передній служать по сумісництву вентиляційними гратами. 60-міліметровий кулер на задній стінці спочатку працює на максимальних обертах, поводячись досить галасливо, але після установки ОС затихає до прийнятного рівня. Передбачене тільки одне робоче положення корпуса – на бічному ребрі, але з обліком пристойної його ширини, воно цілком стійко. Коштує він на товстих гумових ніжках, що додатково підвищують стійкість, а також добре поглинаючий шум і вібрацію. Однак назвати його повністю безшумним не можна – усередині перебуває голосний зумер, що видає пронизливий писк після включення живлення, у момент закінчення завантаження, і перед відключенням живлення, а також у деяких інших

ситуаціях. Включення живлення здійснюється плоскою кнопкою на передній панелі, підсвіченої синім світлодіодом. Для розширення доступної ємності передбачені два порти USB 3.0 і один eSATA (всі вони розташовані позаду). На передній панелі присутня кнопка включення з яскравим синім світлодіодом. Живлення здійснюється від зовнішнього адаптера, виготовленого з гарним запасом потужності, що вдвічі перевищує таку в пристрій Buffalo. У розетку він підключається коротким відрізком кабелю з рознімання IEC 320.

Програмна оболонка Synology теж уніфікована для використання у всіх мережних накопичувачах, незалежно від їхнього калібру, і зветься DiskStation Manager. Недавно вийшла нова версія 5.0, що підняла функціональність і юзабіліті на нову висоту – але з нею, схоже, небагато переборщили. «Мультяшний» інтерфейс із мінімумом налаштувань, де більшість дій виконується «на автоматі» – мрія новачка, але головний біль просунутого адміністратора, що не одержує альтернативних шляхів рішення виниклої проблеми. З однієї з них ми зштовхнулися й самі, при спробі встановити дискову ОС, без якої пристрій попросту непрацездатний.

Процес установки ПЗ починається з відвідування спеціальної сторінки на сайті synology.com, java-скрипт якої прямо із браузера знаходить підключений до мережі накопичувач і запускає завантаження образа DiskStation Manager (перед цим нас настирливо попереджають, що всі дані на диску будуть віддалені, тому якщо там раптом залишилося щось потрібне, прийде скопіювати дані кудись ще, навіть якщо їх там всі 4 терабайта). Після цього ми одержуємо повідомлення про відсутність доступу в мережу, оскільки браузер використовував WAN-підключення через один мережний адаптер, а NAS був приєднаний до LAN через інший, і міст між сегментами мережі був відсутній. Чому не можна було передати файл тим же скриптом, що мав доступ у глобальну мережу? Загадка.

Що ж, вручну завантажуємо файл і вибираємо опцію завантаження з диска. Далі немає ніяких налаштувань і органів керування, лише самотній прогрес-бар з повільно повзучою зеленою смужкою, і систему вдалося

встановити, інакше візит у сервісний центр був би неминучий. Після установки треба кілька екранів з поздоровленнями й пропозиціями зареєструватися на сайті (куди ж без цього), і ми попадаємо в робочий інтерфейс DiskStation Manager, стилізований під робочий стіл Windows 7 з темою Aero. Стиль скопійований дуже ретельно, кожний розділ відкриває своє вікно, що відображається на панелі завдань, є кнопка «Пуск», Панель Керування, навіть вікна при перетаскуванні стають напівпрозорими. У треї перебувають іконки статусу й повідомлень. Якщо є бажання ще більше поліпшити зовнішній вигляд інтерфейсу, можна скачати й установити додаткові теми. Лагів практично не помітно, у тому числі при звертанні до диска, незважаючи на те, що процесор у накопичувачі використовується все той же Marvell Armada 370. Кількість можливостей ПЗ Synology просто вражає: це й підтримка великої кількості протоколів, численні служби й клієнти мережних сервісів. Система підтримує й додаткові програмні модулі, при використанні яких нас змушують прийняти ще одну велику ліцензію.

Продуктивність DS114 перебуває на гідному рівні, хоч і злегка програє виробу Buffalo. Мабуть, причина криється у використовуваному HDD – постав ми той же WD Black, і загублені кілька відсотків удалося б відіграти. Але, по великому рахунку, різницю ви однаково не заметете. Так що ми б поставили йому 10 балів з 10, якби не неадекватна ціна. Вона помітно вище, ніж у конкурента, притім що той продається вже укомплектований диском, а накопичувач Synology порожній! Додамо до його ціни вартість диска, і сума буде вище майже вдвічі. Але, як говориться, ми не настільки багаті, щоб купувати дешеві речі. Зважте всі аргументи: можливо, покупка все-таки коштує того?

WD My Cloud

My Cloud – типово бюджетний пристрій, називаний у деяких інтернет-магазинах навіть не «мережним диском», а просто «зовнішнім диском» (притім, що функцію зовнішнього диска він виконувати не може). Як і пристрій Buffalo, споконвічно він комплектується тридюймовим вінчестером ємністю 2, 3, або 4 Тбайт (у нас виявився перший варіант).

Зовнішній вигляд WD My Cloud нагадує товсту книгу, де корінець є лицьовою стороною. Там є єдиний багатобарвний світлодіод, що показує нам стан пристрою. «Обкладинка» її гладка, а обрізи блоку сторінок поцятковані вентиляційними отворами. Їх багато, тому що охолодження диска повністю пасивне. Єдине передбачене конструкцією положення – стійма на високих гумових ніжках, що забезпечують як безперешкодне надходження повітря усередину, так і надійну віброізоляцію. Живлення він одержує із субтильного адаптера, потужністю всього 18 Вт, але, видимо, цього виявляється досить. Вимикача або кнопки включення в накопичувача ні, і завершення роботи доводиться виконувати через меню веб-інтерфейса. Залогінившись у нього під адміністративним аккаунтом, варто пройти по ланцюжку меню «параметри – службові програми – обслуговування пристрою – завершення роботи». Зробивши це пари раз, у третій повторювати вже не захочеться! Простіше залишити пристрій включеним. Але ми трохи забігли вперед.

Угадайте, диск якого виробника встановлений усередині? Звичайно ж, WD – це такий же WD Red, який використовувався в накопичувачі Synology, хіба що меншого обсягу. Досить тихий сам по собі, укупі з відсутністю вентилятора він перетворює My Cloud в один із самих безшумних пристроїв подібного роду. З портів розширення є тільки один порт USB, але зате версії 3.0. Є присутнім і слот для замка Kensington. Крім нестандартного дизайну, в апарату застосовується також нестандартне схемотехнічне рішення: як процесор використаний майже нікому не відомий So Comcerto 32200 виробництва компанії Mindspeed, поглиненої в грудні минулого року холдингом M/ A-COM Technology Solution. Цей досить сучасний кристал, виготовлений за технологією 40 нм, містить два ядра ARM Cortex-A9, здатних функціонувати на частоті до 1,2 Ггц, але в даному виробі працюючих на 650 Мгц, але оскільки ядер два, а не одне, їхня сумарна продуктивність виявляється приблизно на тім же рівні, що й в інших учасників огляду. А от на оперативній пам'яті відверто заощадили, установивши всього 256 Мбайт.

Користувальницький інтерфейс продукту робить суперечливе враження. Стильна чорно-біла колірна гама контрастує з іконками й шрифтами, що нагадують інтерфейс смартфоної версії Windows 8. На початку роботи нам пропонують вибрати мову (росіянин у списку присутній), і зареєструватися на сайті для того, щоб зробити наша персональна хмара доступним через глобальну Мережу. Потім ми попадаємо на головний екран з основною інформацією про нашу систему й закладками для завдання налаштувань. Угорі є три окремі іконки для виклику короткої довідки, перегляду повідомлень і активності USB-порту. Налаштування досить аскетичні – наприклад, розмір фрейму для Ethernet-Адаптера задати не вийде, але ж це один з основних параметрів, що впливають на пропускну здатність мережі. Набір функцій також невеликий: крім згаданого «персональної хмари», є тільки мультимедійний сервер DLNA, а також унікальна функція SafePoints, що дозволяє робити копії поточного стану диска на випадок раптового збою, аналогічно тому, як це зроблено в ОС Windows. Завдяки мінімуму прикрас, лагів при роботі з інтерфейсів не відчувається.

При тривалій роботі й пасивному охолодженні диск нагрівається досить помітно: напевно, це причина того, що в статусній інформації нам не повідомляють точного значення температури, інформуючи лише є чи чи перегрів ні, хоча статистика дата-центрів говорить про те, що ресурс жорсткого диска залежно від робочої температури може мінятися досить сильно. Тестування продуктивності показало досить скромні результати, що уступають нашим лідерам приблизно на третину. Почасти це можна пояснити неоптимальними налаштуваннями мережі, почасти – меншою частотою процесора. Для переважної більшості застосувань її буде досить, наприклад, для перегляду чотирьох одночасних потоків відео у форматі Full-HD є майже дворазовий запас. З обліком усього перерахованого вище, WD My Cloud можна порекомендувати починаючому користувачеві, що ще не визначився з тим, які ж додаткові можливості він хотів би бачити у свого накопичувача, і бажаючий за мінімальні гроші мати мінімум проблем.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Програмне забезпечення написане мовою Visual C#. Ця мова обрана виходячи з наступних міркувань. Visual C# – строго типізована об'єктно-орієнтована мова, призначена для розробки різноманітних безпечних і потужних додатків, виконуваних у середовищі .NET Framework. Мовою Visual C# можна розробляти звичайні клієнтські додатки Windows, веб-служби XML, розподілені компоненти, додатки типу "сервер-клієнт", додатки баз даних і багато яких інших. В Visual C# 2012 є розширений редактор коду, конструктори зі зручним користувальницьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку додатків мовою Visual C# версії 5.0 і .NET Framework версії 4.5.

Синтаксис Visual C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C++ або Java з легкістю взнають синтаксис із фігурними дужками, характерний для мови Visual C#. Розроблювачі, що знають кожну із цих мів, як правило, зможуть домогтися ефективної роботи з мовою Visual C# за дуже короткий час. Синтаксис Visual C# робить простіше те, що було складно в C++, і забезпечує потужні можливості, такі як типи значень Nullable, перерахування, делегати, лямбда-вираження й прямий доступ до пам'яті, чого немає в Java. Visual C# підтримує універсальні методи й типи, забезпечуючи більше високий рівень безпеки й продуктивності, а також ітератори, що дозволяють при реалізації колекцій класів визначати власне поводження ітерації, що може легко використовуватися в клієнтському коді. В Visual C# 5.0 вираження LINQ (Language-Integrated Query) роблять строго-типізований запит першокласною конструкцією мови.

Як об'єктно-орієнтована мова, Visual C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод `Main` – крапку входу додатка – інкапсулюється у визначення класів. Клас може

успадковувати безпосередньо з одного родового класу, але може реалізовувати будь-яке число інтерфейсів. Для методів, які перевизначають віртуальні методи в батьківському класі, необхідно ключове слово override, щоб виключити випадкове повторне визначення. У мові Visual C# структура схожа на полегшений клас: це тип, що розподіляється по стопках, що реалізує інтерфейси, але не підтримуюче спадкування.

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

– Інкапсульовані підписи методів, називані делегатами, які підтримують строго-типізовані повідомлення про події.

– Властивості, що виступають у ролі методів доступу для закритих змінних-членів.

– Атрибути з декларативними метаданими про типи під час виконання.

– Вбудовані коментарі XML-документації.

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

Якщо буде потрібно забезпечити взаємодію з іншим програмним забезпеченням Windows, таким як об'єкти COM або власні бібліотеки DLL Win32, у мові Visual C# можна використовувати процес, що називається "Interop". Процес Interop дозволяє програмам на Visual C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C++. Мова Visual C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення. Процес побудови Visual C# у порівнянні з C і C++ простий і є більше гнучким, чим в Java. Немає окремих файлів заголовка, а методи й типи не потрібно повідомляти в певному порядку. У вихідному файлі Visual C# може бути визначене будь-яке число класів, структур, інтерфейсів і подій.

Програма мовою Visual C# виконується в середовищі .NET Framework – інтегрованому компоненті Windows, що містить віртуальну систему виконання (середовище CLR) і уніфікований набір бібліотек класів. Середовище CLR являє собою комерційну реалізацію корпорацією Майкрософт інфраструктури CLI, що є міжнародним стандартом, який лежить в основі створення середовищ виконання й розробки, у яких забезпечується тісна взаємодія між мовами й бібліотеками. Вихідний код, написаний мовою Visual C#, компілюється в проміжну мову (IL) у відповідності зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення й рядки, зберігаються на диску 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.

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

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи Cloud-технології для NAS.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності

розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

При виборі мережного сховища компанії керуються такими критеріями, як забезпечувана продуктивність, гарантована цілісність даних, доступність і масштабованість, підтримка резервного копіювання й необхідні засоби керування зберіганням при прийнятній вартості. Спеціалізовані й оптимізовані для різного застосування рішення дозволяють IT-підрозділам домогтися збільшення швидкості архівування даних і наступного доступу до них.

Високопродуктивні системи зберігання, оснащені такими функціями, як кешування даних на SSD, синхронна реплікація й інші засоби «корпоративного класу», тепер по кишені багатьом організаціям. У той же час у складних економічних умовах підприємства середнього бізнесу віддають перевагу пристроям без зайвого функціонала, але намагаються зберегти високий рівень надійності й продуктивності. Всі частіше вони обертають свою увагу на системи зберігання даних, призначені для SMB. Тому вендорам доводиться вибирати таку цінову стратегію, щоб, з одного боку, що відповідають рішення залишалися привабливими для малого бізнесу, а з інший, не далечіні приводу корпоративним замовникам купувати версії для SMB з метою економії. Наприклад, у деяких виробників ціна залежить від числа підтримуваних системою робочих місць.

Замовники прагнуть здобувати більше економічні й ефективні системи для зберігання й захисту даних. Однак компанії із сектора SMB повинні брати до уваги «вартість росту». Покупка СЗД для них – істотні інвестиції. Успішні підприємства малого й середнього бізнесу розширюються постійно, у результаті збільшується кількість необхідних інформаційних сервісів, що, у свою чергу, приводить до додаткових щорічних витрат, підсумкову величину яких не так вуж просто оцінити заздалегідь. Тому дуже важливо, щоб у своїх пропозиціях для

SMB вендори не тільки наголошували на низької вартості первісних вкладень, але й пропонували прозору модель росту при обмежених бюджетах.

Біля року тому Netgear представила нову систему зберігання ReadyNAS 716 (RN716X) з підтримкою 10Gb – «найшвидше у світі мережне сховище в настільному форм-факторі». Її поставки на український ринок почалися цього року. У цій системі, оснащеною пам'яттю ECC ємністю 16 Гбайт, використовується технологія ReadyCLOUD, що дозволяє виконувати віддалене налаштування (із хмари) і управляти файлами. ReadyNAS 716 підтримує жорсткі диски SATA, SSD і має три порти розширення eSATA для масштабування ємності (до 84 Тбайт). Сервіс ReadyNAS Vault автоматично зберігає резервні копії в ЦОД Netgear (у хмарі), а за допомогою опції ReadyNAS Replicate користувачі можуть за допомогою єдиної консолі управляти реплікацією на кілька площадок. Убудована функція ReadyNAS Remote забезпечує безпечний віддалений доступ до даних по VPN.

NVR для відеокамер Axis

Buffalo Technology представила у вересні чергову новинку – TeraStation 5200 Network Video Recorder (NVR). Цей мережний відеореєстратор зі спеціальним прошиванням для Axis Camera Companion забезпечить ефективну установку й керування мережами IP-відеоспостереження в SMB. Buffalo і Axis об'єднали свої зусилля для створення універсальної й продуктивної мережі IP-відеоспостереження, легкої в налаштуванні й простій у керуванні. При цьому Buffalo надає технічну підтримку для всього рішення. Програмне забезпечення управляє камерами, процесами зберігання й запису, у тому числі переглядом і експортом відео. Buffalo TeraStation NVR може записувати й архівувати відео з 16 мережних камер Axis.

Dell Storage SC4020 – універсальний дисковий масив, націлений на ринок SMB, масштабується від 6 до 120 дисків. Основна особливість цього масиву з підтримкою Fibre Channel або iSCSI полягає в тім, що, незважаючи на свою компактність (від 2U) і низьку вартість, він використовує ОС сімейства Dell

Compellent. У результаті підприємства SMB одержують доступ до функціонала старших масивів корпоративного класу, включаючи віртуалізацію ємності зберігання, автоматичне переміщення даних між рівнями зберігання й знімки даних. Уніфікована система підтримує роботу із флеш-пам'яттю, у тому числі в гібридній і бездисковій конфігураціях, а також інтеграцію з VMware. До того ж її продуктивність підвищується завдяки автоматичному налаштуванню, що виконується на основі інформації про використання, одержуваної в режимі реального часу.

EMC VNXe 3200 ставиться до систем зберігання, призначеним для ринку SMB, але має функціональність систем корпоративного класу. На додаток до традиційного для NAS файловим сервісам вона надає блоковий доступ по протоколі Fibre Channel, підтримує флеш-накопичувачі, дозволяє розширювати кеш-пам'ять масиву шляхом додавання флеш-пам'яті в режимі читання й запису, а також надає кошти для організації багаторівневого зберігання з автоматичним переміщенням блоків даних між рівнями. Крім того, система зберігання VNXe 3200 оптимізована для сучасних багатоядерних процесорів. Корпоративних замовників цей продукт зацікавить своєю невисокою вартістю, а замовників з SMB – корисними можливостями.

3.2 Розробка структурної схеми

На українському ринку добре відомі такі постачальники систем NAS для SMB, як Buffalo, D-Link, LenovoEMC, Seagate, Netgear, Synology, Thecus, QNAP і WD. Вони регулярно обновляють свої лінійки продуктів, розширюють їх як «нагору», випускаючи могутніші багатодискові системи для великих компаній, так і «долілиць», доповнюючи спектр пропозицій рішеннями для будинку й малого офісу (SOHO), а також для різних спеціалізованих завдань. Не забувають про цей сегмент і виробники «великих» систем зберігання, наділяючи їхнім перевіреним функціоналом старших моделей СЗД.

NAS для резервного копіювання

Консолідація даних спрощує їхній захист, тому резервне копіювання даних – одна з найважливіших функцій мережних сховищ, причому деякі з них оптимізовані для виконання саме цього завдання. Програмне забезпечення, що розроблене в ході виконання роботи, кожні 15 хв генерує інкрементні знімки для створення повних резервних копій даних серверів, настільних ПК і ноутбуків. Тепер користувачі одержують можливість самостійно відновлювати файли, папки й цілі бази даних при роботі в будь-якому середовищі – фізичної або віртуальної. У результаті відпадає необхідність у регулярному резервному копіюванні всієї системи. Завдяки файловій системі нового покоління забезпечуються цілісність даних, ефективне використання ємності зберігання й мінімальне навантаження на обчислювальні ресурси.

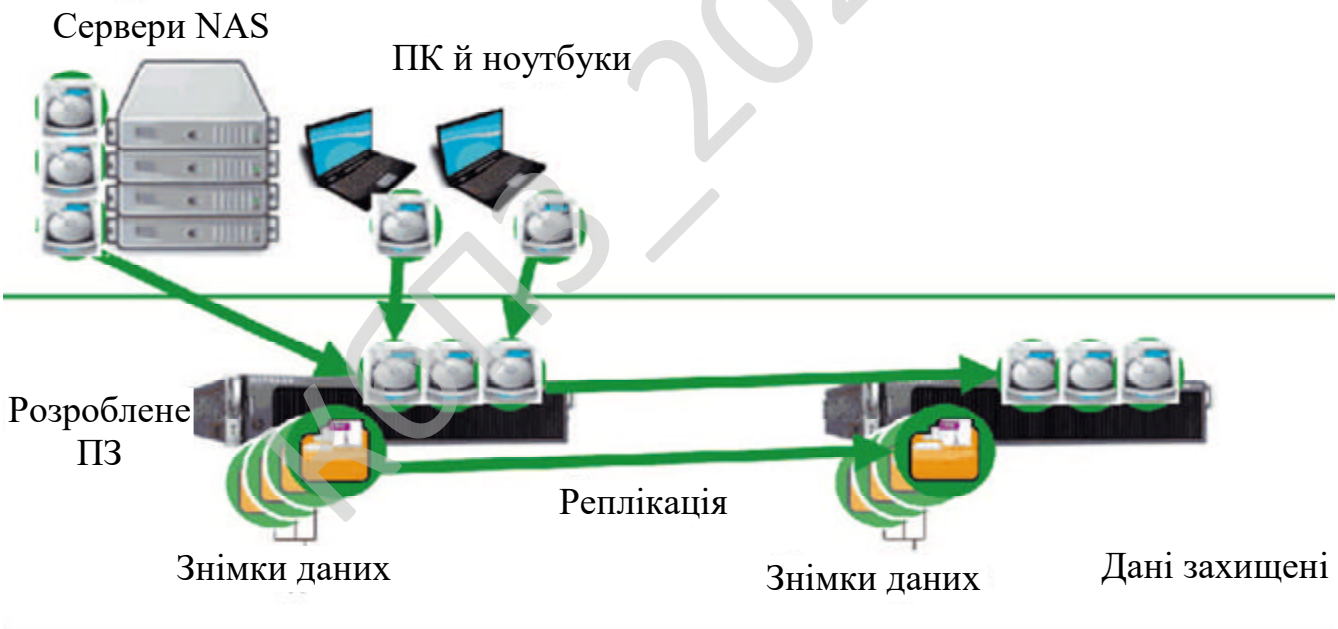

Рисунок 3.1 – Структурна схема системи

Програмне забезпечення, що розроблене в ході виконання роботи, поєднує дискові масиви Netgear ReadyDATA з підтримкою RAID (0, 1, 5, 6, 10) і ПЗ резервного копіювання. Система створює повні резервні копії у форматі VHDX,

які можна використовувати для відновлення даних як у віртуальної, так і у фізичної машинах. Підтримуються будь-які гіпервізори для Windows, у тому числі віртуальні машини Xen, VMware або Hyper-V. Відновлюються навіть окремі об'єкти додатків: наприклад, програмне забезпечення, що розроблене в ході виконання роботи, дозволяє відновити об'єкти Microsoft Exchange (окремі поштові скриньки, папки, повідомлення) без завантаження повної резервної копії.

Це рішення може використовувати кожний знімок як крапка відновлення, фактично являючи собою систему класу near-CDP, але менш дорогу, чим класичні продукти CDP. Замовник позбувається не тільки від надмірних інвестиційних витрат, але й від необхідності одержання спеціальних знань, оскільки й налаштування, і керування надзвичайно прості. Крім того, програмне забезпечення, що розроблене в ході виконання роботи, заощаджує час і ємність, зберігаючи інкрементні знімки даних (записуються тільки змінені дані) з перевіркою резервної копії в реальному часі, і надає всі переваги повних резервних копій, включаючи спрощений процес копіювання й зберігання.

Вартість системи визначається програмною ліцензією, що залежить від числа й типів клієнтів для резервного копіювання, від використовуваної системи ReadyDATA (наприклад, це може бути стійкова версія ReadyDATA 5200 або настільна ReadyDATA 516), видів установлених у ній дисків (SATA/SAS/ NL-SAS/SSD) і умов контракту на підтримку. Цей продукт більш економічний аналогічних рішень Symantec або Unitrends. Крім того, оскільки ємність таких систем резервного копіювання звичайно становить 5-12 Тбайт, у міру її вичерпання доводиться купувати ще один пристрій. Програмне забезпечення, що розроблене в ході виконання роботи, – одне з деяких рішень, масштабованих до 240 Тбайт «сирої» ємності, а число користувачів обмежується лише придбаною ліцензією.

У катастрофостійкій конфігурації з рознесеними системами програмне забезпечення, що розроблене в ході виконання роботи, після створення повної резервної копії здійснюється реплікація змін (інкрементних копій). Тим самим

оптимізується навантаження на канали WAN. Резервне копіювання в хмару поки не передбачено, але Netgear активно працює в цьому напрямку.

Даним рішенням зацікавляться компанії, у яких розгорнуте більше 500 робочих місць. Як правило, вони мають потребу в досить потужному рішенні резервного копіювання й захисту від катастроф (DR), але не хочуть здобувати дорогі й складні системи корпоративного класу. За даними виробника, типовий показник RPO становить 15 хв, а RTO – близько 2 год (залежно від відновлюваного сервера).

Низька вартість і простота використання – основні вимоги до систем такого класу. Наступними по важливості є вартість обслуговування й строки поставки. Нерідко вирішальним фактором вибору стає можливість придбати продукт у магазині або одержати зі складу. Малим компаніям потрібні невеликі системи, що забезпечують простий захист даних від збоїв дисків. Але функціональні вимоги до СЗД ростуть разом з компаніями – у міру підвищення рівня IT-грамотності. Згодом з'являються потреби в розмежуванні доступу до даних, у рольовому адмініструванні, квотуванні надаваної користувачам ємності зберігання, інтеграції із системами керування контентом, резервного копіювання, аудита й антивірусних систем».

Багато сучасні NAS для SMB підтримують досить розвинені функції, серед яких – гнучкий розподіл ресурсів (Thin Provisioning), дедуплікація даних, iSCSI, моментальні знімки, динамічне масштабування JBOD і RAID, реплікація й дзеркалювання томів. Основні розходження між системами NAS різного класу складаються в продуктивності, підтримці тих або інших розширених функцій системи й засобів безпеки, iSCSI і типів RAID, обмеженнях по числу HDD і застосовності системи в доменній структурі. Однак чітко розділити різні класи СЗД стає усе складніше.

Підтримка віртуалізації

Границя між мережними системами зберігання корпоративного класу й NAS для SMB поступово стирається. Це стосується й надійності NAS, і їхньої

продуктивності, і функціональності й масштабованості. Багато ХТО СЗД для SMB мають засобу захисту даних і керування ними, порівнянні із системами більше високого рівня, а деякі NAS інтегруються із хмарними сервісами. Досить затребувана й підтримка віртуалізації серверів. Багато які вендори NAS сертифікують свої системи на сумісність із середовищами віртуалізації VMware, Citrix і Microsoft.

Для ефективної роботи в середовищі віртуалізації системи зберігання даних у першу чергу повинні підтримувати такі технології, як VMware vSphere Storage APIs – Array Integration (VAAI). Крім того, важлива інтеграція на рівні керування: можливість управляти ресурсами зберігання з консолі керування віртуальними середовищами.

Такі NAS повинні мати можливості взаємодії із флеш-пам'яттю, обслуговування великої кількості паралельних запитів до СЗД, підтримки VMware VAAI і Microsoft ODX, а також відпрацьовування відмов шляхів доступу до даних і контролерів СЗД на рівні комутаторів. У СЗД класом вище потрібні такі якості, як віртуалізація портів FC (NPIV, перемикання адрес між контролерами) і взаємодія з гіпервізорами по файлових мережних протоколах, інтеграція механізмів знімків даних з рівнем гіпервізора для забезпечення цілісності даних і зручного відновлення віртуальних машин, а також інтеграція інтерфейсів керування.

Необхідне й достатнє

Що стосується загальної функціональності, деякі вендори вважають, що в SMB NAS обов'язкова наявність мережних інтерфейсів з різними режимами роботи, включаючи резервування, об'єднання й балансування навантаження, підтримку основних рівнів RAID, протоколів SMB/CIFS/SAMBA, AFP, FTP/FTPS, SFTP, HTTP/HTTPS, NFS і розвинених засобів резервного копіювання.

Надійність і захист даних – найважливіші якості NAS. Змушене припинення діяльності й тим більше втрата даних завжди пов'язані з додатковими витратами. До необхідних функцій ставляться також можливості простого й

безпечного віддаленого доступу до даних по мережі і їхньому резервному копіюванню. Однак відмова від повсюдного дублювання компонентів дозволяє тримати цінову планку NAS на прийнятному для SMB рівні, адже ці замовники звичайно не готові платити за рішення з рівнем надійності 99,99%. Наприклад, RAID-Контролер не дублюється, а резервуються тільки блоки живлення.

Ще одним важливим показником є масштабованість – можливість розширення дискового простору. Сучасні технології дозволяють легко нарощувати ємність СЗД, не виключаючи сховище. Не варто забувати й про продуктивність, простоту керування, російськомовній підтримці й строках гарантійного обслуговування встаткування. Багатьом підприємствам може знадобитися додаткова функціональність – наприклад, FTP-сервер, HTTP-сервер, підтримка iSCSI (для середовищ віртуалізації). Однак оптимальне співвідношення ціни й функціональних характеристик залежить від конкретних потреб. Якщо одним компаніям потрібний iSCSI, то іншим досить простого зберігання загальних файлів і резервування даних. Попит на такі функції, як iSCSI target/initiator, Web-сервер, менеджер завантаження й т.д., багато в чому визначається специфікою діяльності компанії.

Для роботи з файлами й передачі даних по мережі системі зберігання потрібні значні обчислювальні ресурси. Основне навантаження лягає на процесор – чим він швидше, тим вище швидкість функціонування всієї системи. Важливе значення має й оперативна пам'ять, особливо при роботі з більшою кількістю дрібних файлів (наприклад, у випадку використання бази даних «1С»). Збільшення обсягів даних приводить і до росту вимог до пропускної здатності корпоративних мереж.

NAS для малого бізнесу повинні оснащуватися необхідними засобами захисту даних, керування їхнім зберіганням і забезпечення безпеки. Такі системи повинні бути максимально простими у використанні (це дозволило б обходитися без залучення IT-фахівців) і легко інтегруватися із застосовуваними додатками. Універсальний сервер NAS – рішення, готове до роботи прямо «з коробки». У

більшості таких систем є утиліти для простої установки й налаштування або підключення інших мережних сховищ, так що допомоги мережного адміністратора не потрібно.

Ролі й функції

NAS можуть використовуватися для самих різних завдань – від FTP- або Web-сервера до відеореєстратора (NVR), причому деякі системи універсальні, а інші являють собою спеціалізовані програмно-апаратні комплекси, розраховані на конкретне застосування й оптимізовані під якесь завдання.

NAS випускаються в стійковому і настільному варіантах і можуть передбачати різні способи розширення ємності й масштабування, тому користувачі одержують практично необмежену ємність. Мережні системи зберігання надають компаніям сегмента SMB зручні можливості централізованого резервного копіювання. Крім того, NAS можуть виконувати різноманітні спеціальні функції – наприклад, застосовуватися як сервер баз даних «1С» із ключем HASP або сервера відеоспостереження. Для підвищення ефективності зберігання в NAS можуть бути убудовані засоби дедуплікації даних. Виробники таких систем, дотримуючись новітніх тенденцій на ринку мережного встаткування, починають оснащувати їх високошвидкісними зовнішніми інтерфейсами 10Gb. Причому мова йде не тільки про старші моделі, розрахованих на обслуговування великої кількості клієнтів, – з 10-гігабітними мережними адаптерами випускаються навіть настільні NAS.

Зараз у молодші моделі СЗД активно переносяться автоматичні класифікація й переміщення даних між різними типами дисків (tiering) – технологія, характерна для старших систем корпоративного класу. Вона дозволяє помітно заощадити на вартості ємності зберігання й збільшити продуктивність. Ще одна важлива функція – знімки даних. Через обмежені бюджети на резервне копіювання технологія миттєвих знімків часто застосовується як заміна резервному копіюванню (що технічно неправильно, але нерідко використовується як змушена міра) або як спосіб для більше гранулярного

відновлення, що доповнює класичну систему резервного копіювання. Завдяки цьому користувачі можуть швидко відновлювати втрачені або віддалені файли.

Більше ємності

За останній рік ємність дискових накопичувачів переборола планку 6 Тбайт. Компанія HGST, що належить Western Digital, представила HDD HGST Ultrastar He6 6 TB із сімома пластинами, що володіють саме такою загальною ємністю, а пізніше випустила 10-терабайтну модель. За рахунок наповнення герметичного корпуса накопичувача гелієм удалося зменшити відстань між пластинами. Seagate випустила нові дискові накопичувачі Seagate 6TB Enterprise 3.5 HDD v4 – «найшвидші у світі жорсткі диски ємністю 6 Тбайт». За інформацією виробника, вони відрізняються надійністю корпоративного класу й оптимізовані для напівоперативного завантаження в режимі 24×7.

Лінійка WD Red поповнилася модифікаціями ємністю 5 і 6 Тбайт, а також моделлю WD Red Pro ємністю 2-4 Тбайт, призначеної для середнього й великого бізнесу. Нова версія технології WD NASware 3.0 дозволяє використовувати HDD у мережних сховищах з підтримкою до восьми відсіків без втрати продуктивності. Накопичувачі WD Red Pro можна застосовувати в системах NAS, що вміщають до 16 дисків.

Флеш-пам'ять і SSD

Вендори NAS акцентують увагу потенційних замовників на різній функціональності, причому можливості, властиві більше дорогим і потужним стійковим сховищам, поступово поширюються й на пристрої молодшого класу. Не стали виключенням флеш-пам'ять і SSD – більшість вендорів уже пропонують таку опцію. «Революція SSD» (за прогнозом IDC, в 2018 році в 80% всіх СЗД будуть застосовуватися флеш-накопичувачі), що відбувається в СЗД корпоративного класу, тепер охоплює й SMB NAS.

Флеш-накопичувачі в системах зберігання даних для ринку SMB з'являються як відповідь на бажання користувачів одержати від своїх систем більше. Вони служать для розміщення даних від додатків з високими вимогами

до продуктивності систем зберігання, наприклад СУБД. Як правило, флеш-накопичувачі встановлюються на додаток до традиційних дисків (гібридні СЗД), але недавно й на цьому ринку стали з'являтися спеціалізовані масиви, що містять тільки флеш-пам’ять. Такі масиви побудовані на новій архітектурі, оптимізованої для роботи із флеш-пам'яттю, займають мало місця й відрізняються низьким рівнем енергоспоживання.

Головна архітектурна відмінність подібних систем – можливість розпаралелити внутрішні звертання до дисків і обробляти за допомогою контролера більше число операцій одночасно. В SSD є дві принципових переваги: малий час відгуку й можливість істотно збільшити чергу до накопичувача без втрати в часі відгуку. Для їхньої реалізації необхідно переглянути внутрішні механізми роботи контролерів з накопичувачами. З SSD зв'язані й деякі інші зміни в СЗД – наприклад, використання кеш-пам'яті запису (у деяких випадку SSD застосовуються як розширення кеш-пам'яті запису, а в інші – як її заміна).

Флеш-пам'ять у недорогих масивах для SMB ставиться до числа інновацій, на практиці які доказали свою цінність для SMB і широко затребуваних. Твердотільні накопичувачі підвищують ефективність роботи з віртуальними середовищами (підприємства із сегмента SMB зараз активно освоюють віртуалізацію й успішно застосовують її), а крім того, забезпечують реальну економію в таких технічно складних проектах, як VDI. У сполученні з автоматичним розподілом даних по типах дисків вони дозволяють підвищити продуктивність інфраструктури зберігання й не вийти за рамки розумного бюджету. Приклади таких систем – Dell PS6210XS (гібридна СЗД із SSD і HDD SAS 10K, застосовувана при віртуалізації серверів і для VDI до 1000 робочих місць) і Dell SC4020 (універсальний масив FC/iSCSI/NFS/CIFS з конфігурацією на флеш-пам'яті – All-Flash). Ємність останнього становить 12 Тбайт і більше, він придатний, у числі іншого, для великих середовищ віртуалізації й для систем VDI, що обслуговують десятки тисяч робочих місць.

Кількість і якість

З удосконалюванням програмного забезпечення систем зберігання підвищується ефективність використання NAS в SMB, поліпшуються продуктивність, сумісність і безпека. Засоби адміністрування СЗД для малого бізнесу теж стрімко еволюціонують: різні операції, наприклад створення тому, сталі виконуватися набагато простіше й швидше. Крім того, за останні кілька років були вдосконалені функції виводу звітів і моніторингу, дуже корисні, зокрема, для планування покупки додаткових СЗД із метою збільшення ємності зберігання.

Системи зберігання будуть розвиватися як кількісно (збільшення ємності дисків, більше число дисків, нарощування кеш-пам'яті й портів уведення-виводу), так і якісно (функції, характерні для корпоративних систем). Наділення систем зберігання для ринку SMB функціями «корпоративного класу» можна вважати стійкою тенденцією. Варто також очікувати розвитку інтеграції із хмарними сервісами й із системами віртуалізації. У складних економічних умовах оптимальне співвідношення ціни й функціональних характеристик робить «малі» системи NAS привабливими для замовників самого різного профілю.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Для підвищення стійкості інформації, яка зберігається в NAS, використовується кодер БЧХ. Процедура кодування інформації, яка зберігається у Cloud-технології для NAS виконується щораз при записі нових даних на диск. Зниження швидкодії обумовлене необхідністю виконання додаткових операцій над даними. Тому вдосконалення й адаптація алгоритму кодування повинне проводитися убік зменшення часу кодування, пропорційного кількості дискових операцій $N_{кдо}$ (більше значимий критерій) і інструкцій процесора (ключових операцій) $N_{кко}$ (менш значимий).

Процедура виправлення помилок виконується тільки при виникненні помилки і є відносно рідкою подією. Тому, у роботі запропонований як найбільше значимий критерій вибрати реалізовану здатність, що виправляє, алгоритму S, а інші критерії, що мають меншу значимість – кількість дискових операцій N_{ddo} й кількість команд (ключових операцій) N_{dko} при декодуванні, пропонується оптимізувати в останню чергу.

1. У роботі була рекомендована наступна поетапна методика вибору ефективного алгоритму кодування СНЗІ:

– Досліджувати шляхи зменшення кількості дискових операцій при різних варіантах запису кодових груп N_{kdo}, скласти список можливих алгоритмів.

– Для кожного варіанта визначити чисельні значення критерію $N_{kdo.}$ Вибрати зі списку алгоритм, що забезпечує мінімальне N_{kdo}.

– Досліджувати шляхи зменшення числа ключових операцій, необхідних для обчислення перевірочних розрядів N_{kko}, скласти список можливих варіантів реалізацій алгоритмів.

– Визначити чисельні значення критерію N_{kdo} Вибрати варіант (сукупність варіантів) реалізації алгоритму, що дозволяє мінімізувати параметр N_{kko}.

2. Рекомендована поетапна методика вибору ефективного алгоритму декодування:

– Досліджувати методи максимізації реалізованої здатності, що виправляє, S, скласти список можливих алгоритмів.

– Визначити чисельні значення критерію S. Вибрати зі списку алгоритм із найкращим значенням S як основу алгоритму декодування двомірного ітеративного коду СНЗІ.

– Досліджувати варіанти реалізації, що дозволяють прискорити процес виправлення помилок по метриках N_{ddo} й N_{dko}, скласти список можливих варіантів реалізацій алгоритмів.

– Визначити чисельні значення критерію $N_{ддо}$ й $N_{дко}$ Вибрати варіант (сукупність варіантів) реалізації алгоритму, що дозволяє мінімізувати параметри $N_{ддо}$ й $N_{дко}$.

Для визначення значень кількості дискових операцій використовувався звичайний підрахунок кількості зчитувальних-записуваних секторів, необхідних для виконання операцій запису даних і відновлення інформації, яка зберігається у Cloud-технології для NAS.

Критерій S (реалізована здатність, що виправляє) обчислювався відповідно до формул теорії завадостійкого кодування.

Вибір методики визначення чисельних значень метрик критеріїв $N_{кко}$ й $N_{дко}$ проводився відповідно до вимог: ступінь адекватності поставленому завданню одержуваних за допомогою обчислювальної моделі оцінок, трудомісткість подання алгоритмів і незалежність моделі від архітектури й механізмів керування ресурсами операційної системи. Були оцінені наступні обчислювальні моделі й методи оцінки обчислювальної складності:

– Машини Тьюринга.

– Метод зведення завдання.

– Метод оцінки складності з використанням граф-схем алгоритмів.

Показано, що метод побудови граф-схем алгоритмів, є єдиним методом, що дозволяє одержати точні значення критеріїв $N_{кко}$ й $N_{дко}$, і описати дрібні відмінності близьких за структурою алгоритмів, тому в роботі був обраний даний метод.

Розглянемо більш детально кожний з функціональних блоків розробленої програми.

Програма записує дані на носій інформації, яка зберігається у Cloud-технології для NAS піддаючи їх кодуванню за методом БЧХ. Це робиться для того, щоб при декодуванні, якщо виникла помилка, то цю помилку можливо було локалізувати та виправити.

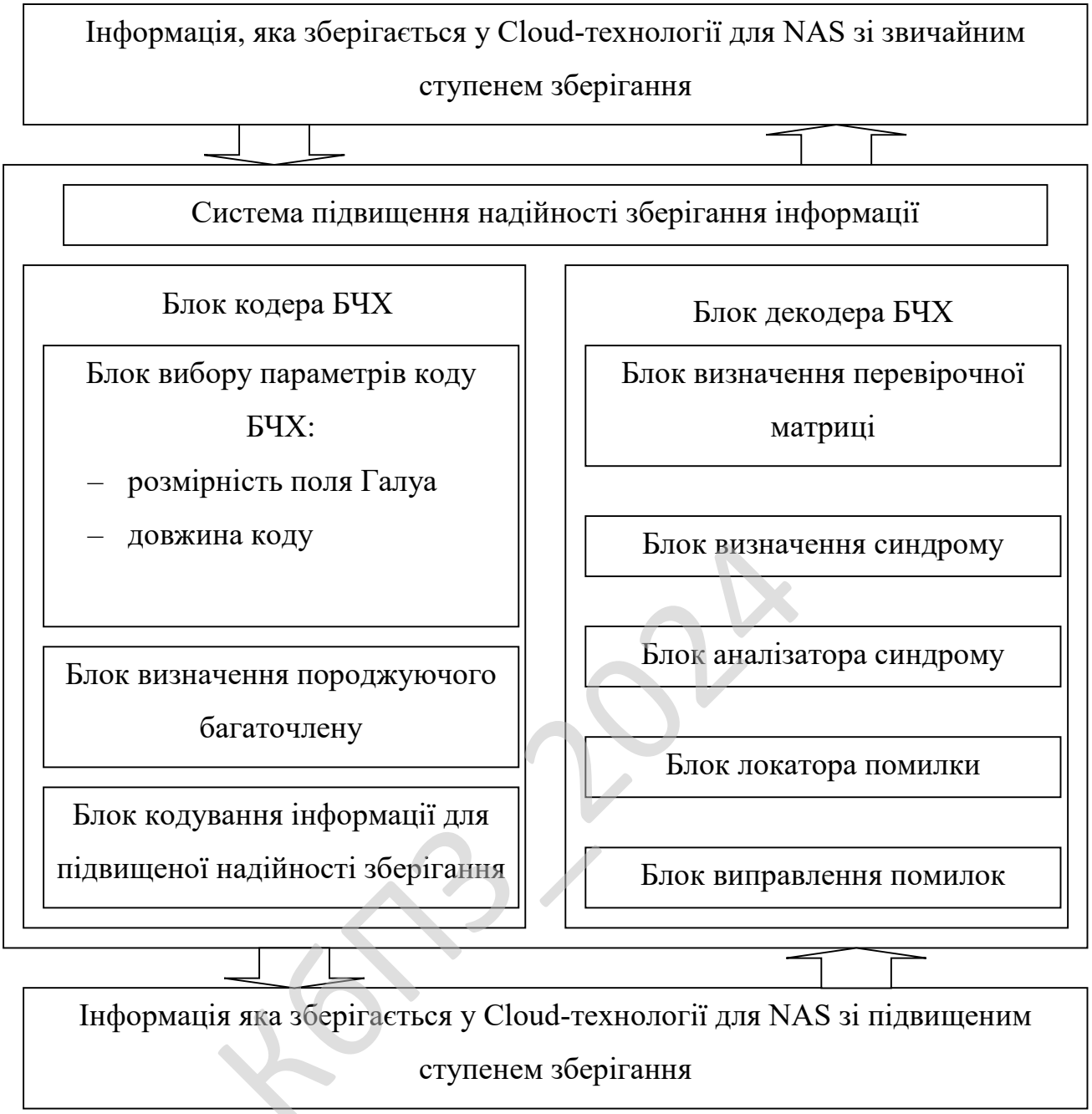

Рисунок 3.2 – Функціональна схема системи

Незакодована інформація подається на блок кодування, де відбуваються наступні дії:

1. Вибираються наступні параметри коду БЧХ, для проведення операції оптимального кодування, згідно з розміром даних:

– розмірність поля Галуа;

– довжина коду;

– мінімальна кодова відстань.

2. Визначається породжуючий багаточлен.

3. Відбувається кодування інформації, яка зберігається у Cloud-технології для NAS для підвищеної надійності зберігання

Після виконання цих дій, інформація зберігається у закодованому вигляді, що дає змогу підвищити рівень її надійності зберігання.

Для того, що прочитати інформацію вона поступає на декодер кодів БЧХ, який виконує наступні дії:

1. Згідно параметрів кодування відбувається визначення перевірочної матриці.

2. Розраховується синдром помилок.

3. Проводиться аналізатор синдрому.

4. Якщо є помилка то вона локалізується.

5. Якщо помилки є то вони виправляються.

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

3.4 Розробка діаграми процесів

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

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає

уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

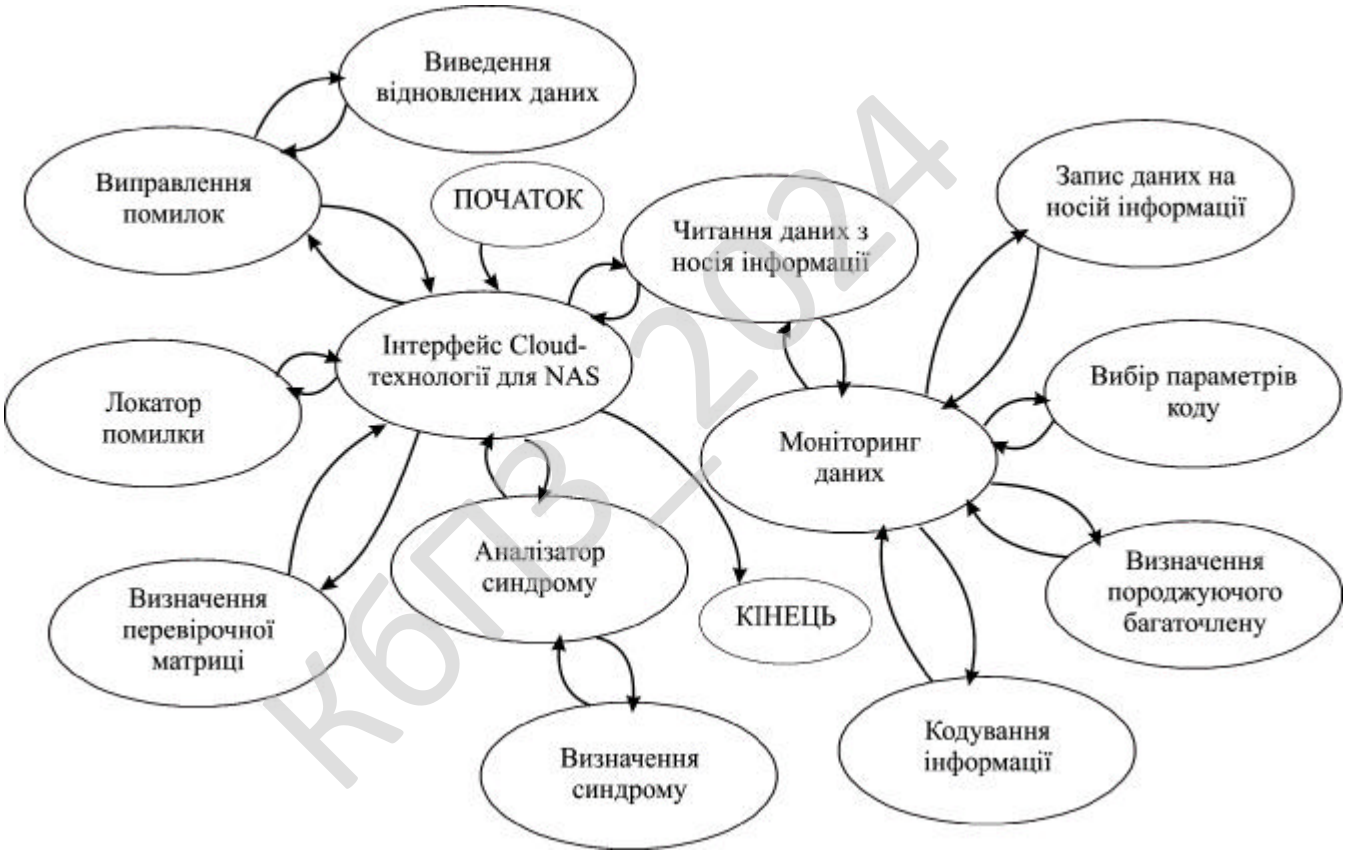

Рисунок 3.3 – Діаграма взаємодії процесів

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

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

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

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

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

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

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

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

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

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

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

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

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

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки

програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю Cloud-технології для NAS.

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

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

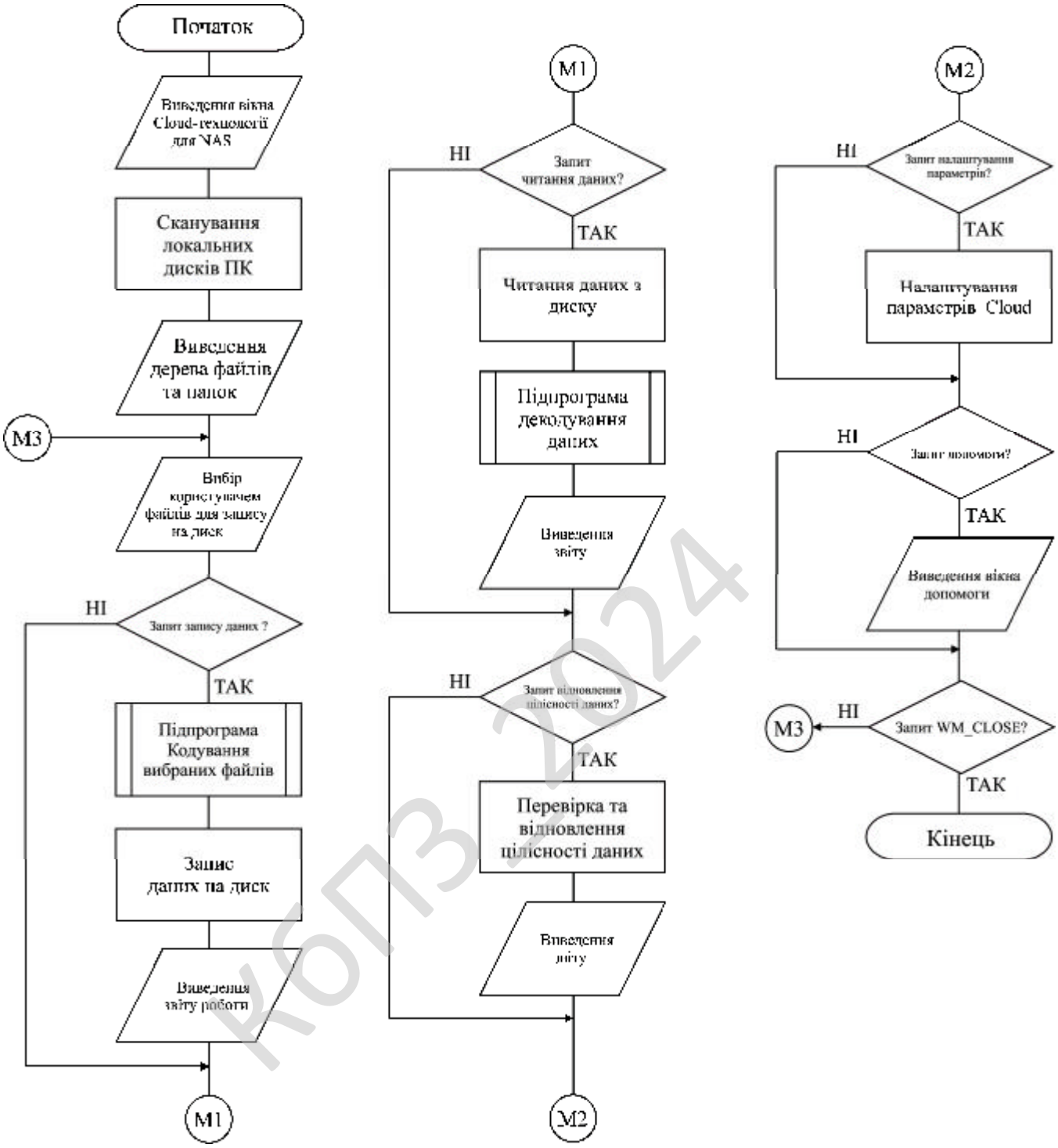

Рисунок 4.1 – Блок-схема основної програми

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

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

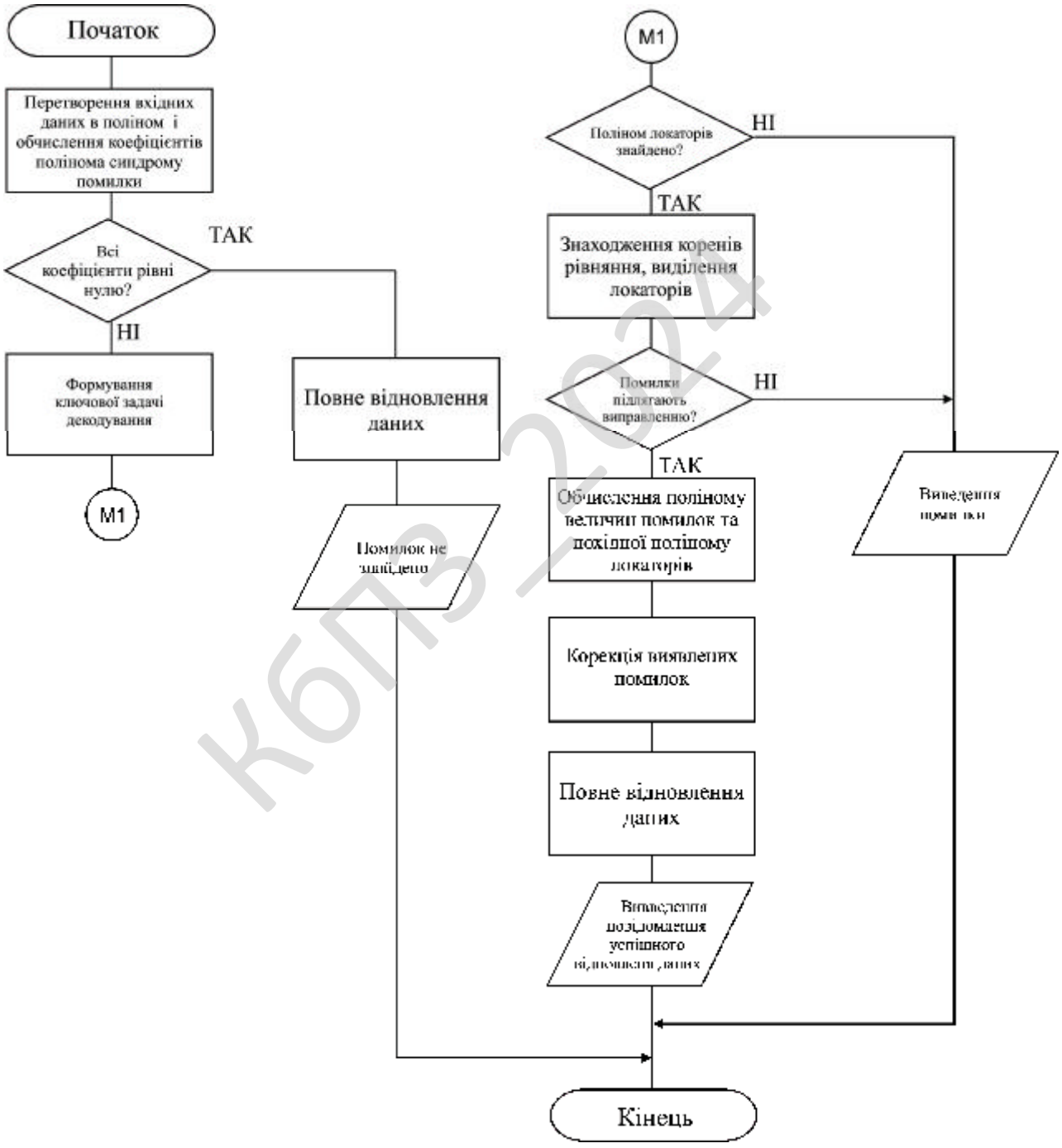

Рисунок 4.2 – Блок-схема роботи підпрограми

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

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

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

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

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

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

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

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

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

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

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

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

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

Алгоритм Берклемпа-Мессі, працює наступним чином. Задати необхідну послідовність біт:

$$s_0, s_1, ..., s_{n-1}.$$

Створити масиви b, t, cдовжини n, задати початкові значення:

$$b_0 \leftarrow 1, c_0 \leftarrow 1, N \leftarrow 1, L \leftarrow 0, m \leftarrow -1.$$

Поки $N < n$:

1. Обчислити:

$$d \leftarrow s_N \oplus c_1 s_{N-1} \oplus c_2 s_{N-2} \oplus ... \oplus c_L s_{N-L}.$$

2. Якщо $d = 0$, то поточна функція генерує обрану ділянку:

$$s_{N-L}, s_{N-L+1}, \ldots, s_N$$

послідовності; залишити функцію колишньої.

3. Якщо $d \neq 0$:

– Зберегти копію масиву cв.t

– Обчислити нові значення:

$$c_{N-m} \leftarrow c_{N-m} \oplus b_0, c_{N-m+1} \leftarrow c_{N-m+1} \oplus b_1, \ldots, c_{n-1} \leftarrow c_{n-1} \oplus b_{n-N+m-1}.$$

– Якщо $2L \leqslant N$, установити значення:

$$L \leftarrow N + 1 - L, m \leftarrow N,$$

і скопіювати t в b.

4. $N \leftarrow N + 1$.

У результаті масив c – функція зворотного зв'язка, тобто:

$$c_L s_i \oplus c_{L-1} s_{i+1} \oplus c_{L-2} s_{i+2} \oplus \ldots \oplus c_0 s_{i+L} = 0,$$

для будь-яких i.

Розглянувши алгоритми, наведемо програмну реалізацію кодера та декодера кодів БЧХ, реалізовану на мові Visual C#. Програмна реалізація кодера БЧХ.

```
_BCH()
{
        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;
              }
       }
}
```

Програмна реалізація декодера БЧХ.

```
_BCH()
{
       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 ламбла
// 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)
                  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 виявлене
        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

Захист розробленого програмного забезпечення буде відбуватися за допомогою IDEA – симетричний блоковий алгоритм шифрування даних, запатентований швейцарською фірмою Ascom. Відомий тим, що застосовувався в пакеті програм шифрування PGP. У листопаді 2000 року IDEA був представлений

як кандидат у проекті NESSIE в рамках програми Європейської комісії IST (англ. Information Societes Technology, інформаційні громадські технології).

Першу версію алгоритму розробили в 1990 році Лай Сюецзя (Xuejia Lai) і Джеймс Мессі (James Massey) зі Швейцарського інституту ETH Zürich (за контрактом з Hasler Foundation, яка пізніше влилася в Ascom-Tech AG) як заміна DES (англ. Data Encryption Standard, стандарт шифрування даних) і назвали її PES (англ. Proposed Encryption Standard, запропонований стандарт шифрування). Потім, після публікації робіт Біхамом і Шаміра по диференціальному криптоанализу PES, алгоритм був поліпшений з метою посилення криптостійкості і названий IPES (англ. Improved Proposed Encryption Standard, покращений запропонований стандарт шифрування). Через рік його перейменували в IDEA (англ. International Data Encryption Algorythm).

Так як IDEA використовує 128-бітний ключ і 64-бітний розмір блоку, відкритий текст розбивається на блоки по 64 біт. Якщо таке розбиття неможливо, останній блок доповнюється різними способами певною послідовністю біт. Для уникнення витоку інформації про кожному окремому блоці використовуються різні режими шифрування. Кожен вихідний незашифрований 64 – біт ний блок ділиться на чотири підблока по 16 біт кожен, так як всі алгебраїчні операції, що використовуються в процесі шифрування, відбуваються над 16-бітними числами. Для шифрування і розшифрування IDEA використовує один і той же алгоритм.

Позначення операцій:

- $\boxplus$ Додавання за модулем 2^{16}.
- $\odot$ Множення за модулем $2^{16}+1$.
- $\oplus$ Побітова виключна диз'юнкція.

Фундаментальним нововведенням в алгоритмі є використання операцій з різних алгебраїчних груп, а саме:

Додавання за модулем 2^{16}.

Множення за модулем $2^{16}+1$.

Побітова виключна диз'юнкція (XOR).

Ці три операції несумісні в тому сенсі, що ніякі дві з них не задовольняють дистрибутивному закону, тобто:

$$a \odot (b \oplus c) \neq (a \odot b) \oplus (a \odot c).$$

Застосування цих трьох операцій ускладнює криптоаналіз IDEA в порівнянні з DES, який базується виключно на операції виключає АБО, а також дозволяє відмовитися від використання S-блоків і таблиць заміни. IDEA є модифікацією мережі Фейстеля.

Генерація ключів

З 128-бітного ключа для кожного з восьми раундів шифрування генерується по шість 16-бітних подключів, а для вихідного перетворення генерується чотири 16-бітних підключа. Всього буде потрібно 52 = 8 x 6 + 4 різних подключів по 16 біт кожен. Процес генерації п'ятдесяти двох 16-бітних ключів полягає в наступному:

Насамперед, 128-бітний ключ розбивається на вісім 16-бітних блоків. Це будуть перші вісім подключів по 16 біт кожен – $(K_1^{(1)}K_2^{(1)}K_3^{(1)}K_4^{(1)}K_5^{(1)}K_6^{(1)}K_1^{(2)}K_2^{(2)})$

Потім цей 128-бітний ключ циклічно зсувається вліво на 25 позицій, після чого новий 128-бітний блок знову розбивається на вісім 16-бітних блоків. Це вже наступні вісім подключів по 16 біт кожен – $(K_3^{(2)}K_4^{(2)}K_5^{(2)}K_6^{(2)}K_1^{(3)}K_2^{(3)}K_3^{(3)}K_4^{(3)})$

Процедура циклічного зсуву і розбивки на блоки триває до тих пір, поки не будуть згенеровані всі 52 16-бітних підключа.

Шифрування

Структура алгоритму IDEA показана на рисунку 4.3.

Процес шифрування складається з восьми однакових раундів шифрування і одного вихідного перетворення. Вихідний незашифрований текст ділиться на блоки по 64 біта. Кожен такий блок ділиться на чотири підблока по 16 біт кожен. На рисунку ці підблоки позначені D_1, D_2, D_3, D_4. У кожному раунді використовуються свої підключі згідно з таблицею підключів.

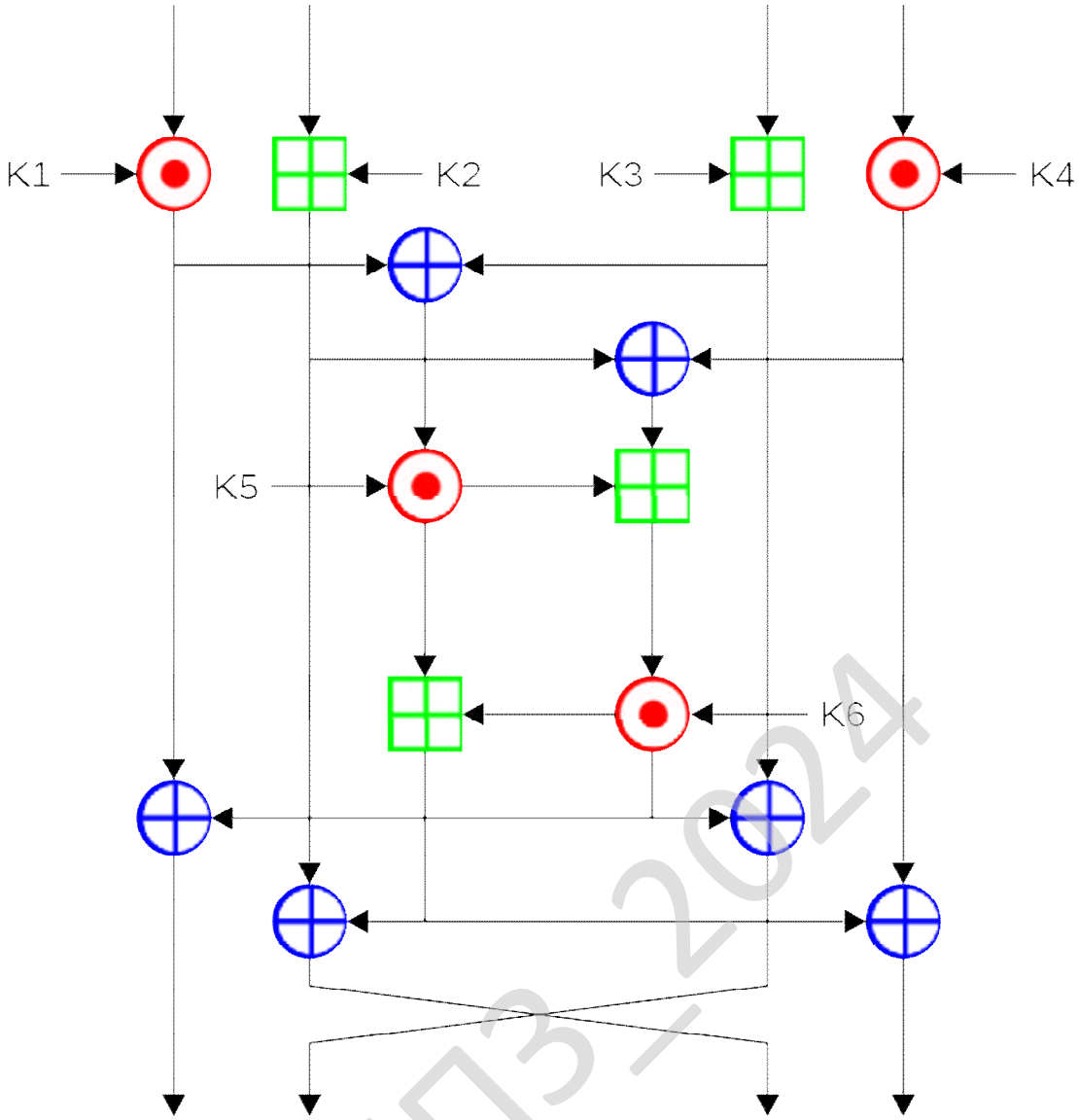

Рисунок 4.3 – Структура алгоритму IDEA

Над 16-бітними підключами і підблока незашифрованого тексту проводяться наступні операції:

– Множення за модулем $2^{16}+1 = 65537$, причому замість нуля використовується 2^{16}.

– Додавання за модулем 2^{16}.

– Побітове виключне АБО.

В кінці кожного раунду шифрування є чотири 16-бітних підблоки, які потім використовуються як вхідні підблоки для наступного раунду шифрування.

Вихідна перетворення являє собою скорочений раунд, а саме, чотири 16-бітних підблоки на виході восьмого раунду і чотири відповідних підключа піддаються операціям:

– Множення за модулем $2^{16}+1$.

– Додавання за модулем 2^{16}.

Після виконання вихідного перетворення конкатенація підблоків D_1', D_2', D_3' і D_4' являє собою зашифрований текст. Потім береться наступний 64-бітний блок незашифрованого тексту і алгоритм шифрування повторюється. Так продовжується до тих пір, поки не зашифрують всі 64-бітові блоки вихідного тексту.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи. Розроблене програмне забезпечення Cloud-технології для NAS складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Команди; Операції; Параметри; Довідка.

– Вікна обрання файлів завантаження на основі Cloud-технології для NAS.

– Вікно виведення результату роботи системи з Cloud системою.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Налаштування параметрів роботи розробленого програмного забезпечення та Cloud-системи.

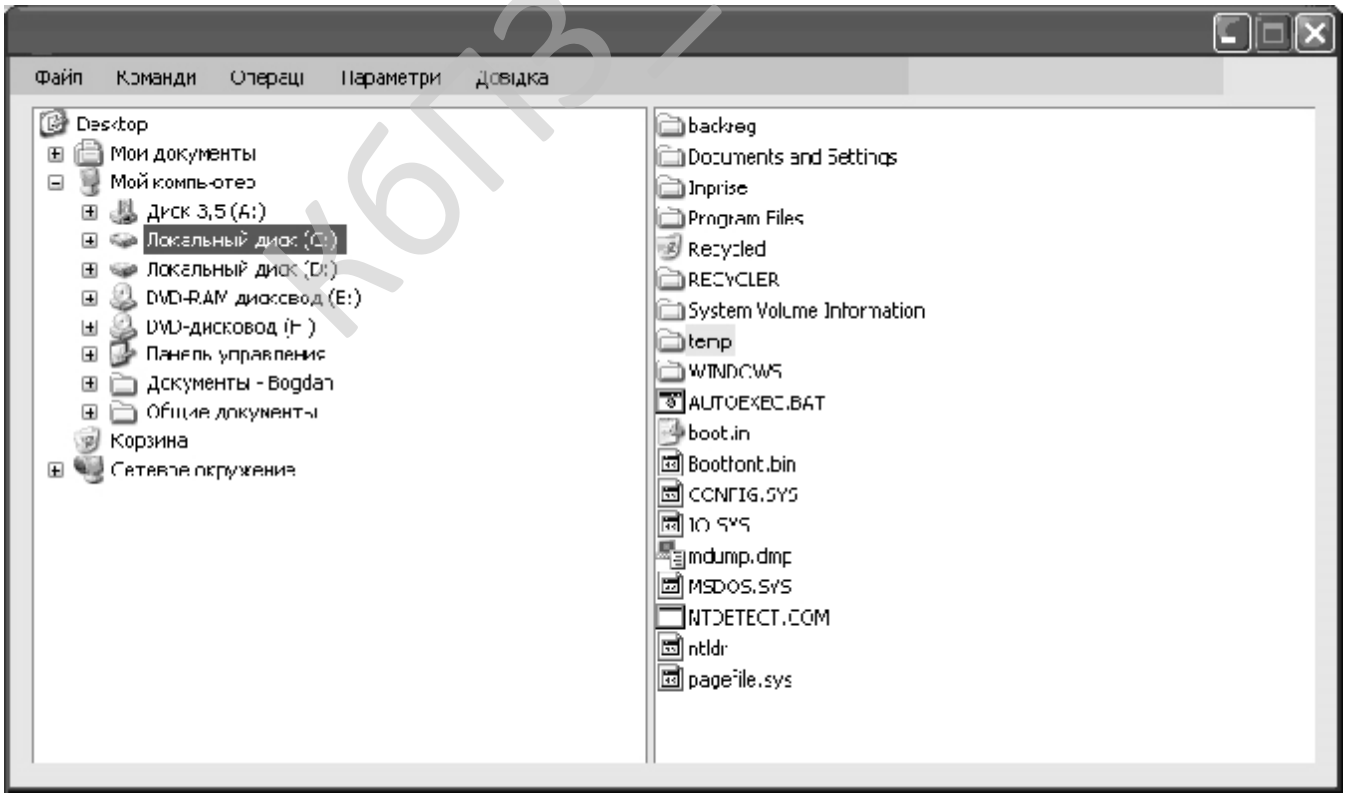

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

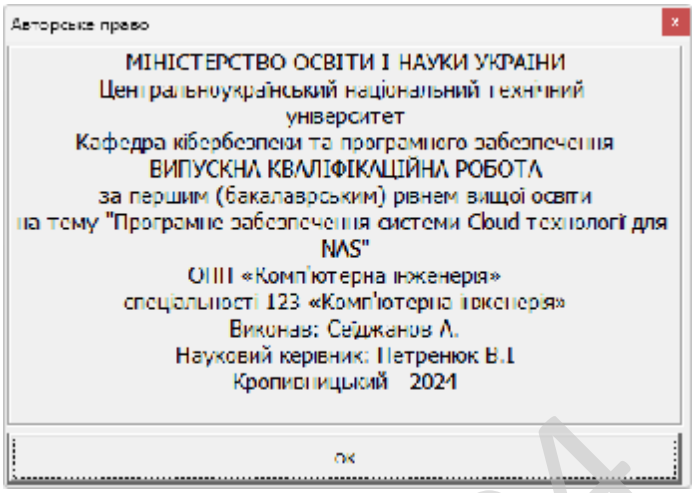

Рисунок 5.2 – Вікно розробника ПЗ

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

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

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

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

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

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

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

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

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

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм.

Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям.

Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм.

Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником.

Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи Cloud-технології для NAS.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем Cloud-технології для NAS.

– Досліджена система Cloud-технології для NAS.

– На основі отриманих результатів досліджень створена програмна реалізація системи Cloud-технології для NAS.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання Cloud-технології для NAS.

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 Cloud-технології для NAS. 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 й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм IDEA.

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. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

2. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

3. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

4. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

5. 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,

6. 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.

7. 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

8. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems:

Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

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

10. 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.

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

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

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.

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

15. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable

Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

16. 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.

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.

18. 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.

19. 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.

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

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

22. 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.

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

24. 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.

25. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

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

27. 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.

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

29. 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.

30. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

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

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

33. 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.

34. 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.

35. 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.

36. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

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

38. 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).

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

40. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

41. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

42. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

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

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

45. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник тез VI міжнародної

науково-практичної конференції “Проблеми та перспективи розвитку ІТ-індустрії”. м. Харків. 17-18 квітня 2014р. – Харків: ХНЄУ. - 2014. - С. 240.

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

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

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

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

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Перелік документів, що розробляються.....5
8 Етапи розробки.....6
9 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи Cloud-технології для NAS.

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. № 131-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи Cloud-технології для NAS.

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 з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи Cloud-технології для NAS;

– 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, 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 – 78 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.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 10.06.2024 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

_____ Петренюк В.І.

Програмне забезпечення системи Cloud-технології для NAS

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 56

Літера: РП

Кропивницький – 2024 року

Файл MainForm.cs - головне вікно програми

```
namespace RecoveryDisk
{
    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("Downloads", 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 = "BChH";
            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 = "Downloads";
            treeNode8.SelectedImageIndex = 20;
            treeNode8.Text = "Downloads";
            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 = "RECYCLER";
            treeNode14.SelectedImageIndex = 20;
            treeNode14.Text = "RECYCLER";
            treeNode15.ImageIndex = 18;
            treeNode15.Name = "System Volume Information";
            treeNode15.SelectedImageIndex = 20;
            treeNode15.Text = "System Volume Information";
            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 ProcessForm.cs - вікно відображення процесів запису/читання та перевірки цілісності даних

```
namespace RecoveryDisk
{
    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 = "Лічільник 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 BCHEncoder.cs - декодер БЧХ Cloud-технології для NAS

```
using System;
using System.Threading;

namespace RecoveryDisk
{
    /// <summary>
    /// Клас кодера БЧХ Cloud-технології для NAS
    /// </summary>
    public class BCHEncoder : BCHBase
    {
        #region Construction & Destruction

        /// <summary>
        /// Конструктор кодера за замовчуванням
        /// </summary>
        public BCHEncoder()
        {
            // 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 BCHEncoder(int dataCount, int eccCount)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, (int)BCHType.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 БЧХ Cloud-технології для NAS (по
типу матриці)</param>
        public BCHEncoder(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 БЧХ Cloud-технології для
NAS (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)BCHType.Dispersal)
            {
                maxVolCount = (int)BCHConst.MaxVolCountDisp;

            } else
            {
                maxVolCount = (int)BCHConst.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.eBCHType)
                    )
                {
                    this.mainConfigChanged = true;
                }

                // Зберігаємо конфігурацію
                this.n = dataCount;
                this.m = eccCount;
                this.eBCHType = 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.eBCHType == (int)BCHType.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.eBCHType == (int)BCHType.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 декодера беремо дані з відповідного
масиву
                    if (this.eBCHType == (int)BCHType.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 (OnBCHMatrixFormingFinish != null)
                {
                    //...повідомляємо, що екземпляр класу готовий до роботи
                    OnBCHMatrixFormingFinish();
                }

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

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

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

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

        #endregion Private Operations
    }
}
```

Файл BCHDecoder.cs - декодер БЧХ Cloud-технології для NAS

```
using System;
using System.Threading;

namespace RecoveryDisk
{
    /// <summary>
    /// Клас декодера БЧХ Cloud-технології для NAS
    /// </summary>
    public class BCHDecoder : BCHBase
    {
        #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 BCHDecoder()
        {
            // 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 BCHDecoder(int dataCount, int eccCount, int[] volList)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, volList, (int)BCHType.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 БЧХ Cloud-технології для NAS (по
типу матриці)</param>
        public BCHDecoder(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 БЧХ Cloud-технології для
NAS (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)BCHType.Dispersal)
            {
                maxVolCount = (int)BCHConst.MaxVolCountDisp;

            } else
            {
                maxVolCount = (int)BCHConst.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.eBCHType)
                    )
                {
                    this.mainConfigChanged = true;
                }

                // Зберігаємо конфігурацію
                this.n = dataCount;
                this.m = eccCount;
                this.eBCHType = 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.eBCHType == (int)BCHType.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
зворотної
                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)
                     &&
                        (OnUpdateBCHMatrixFormingProgress != null)
                   )
                {
                    //...виводимо дані
                    OnUpdateBCHMatrixFormingProgress((((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.eBCHType == (int)BCHType.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.eBCHType == (int)BCHType.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 (OnBCHMatrixFormingFinish != null)
                {
                    //...повідомляємо, що екземпляр класу готовий до роботи
                    OnBCHMatrixFormingFinish();
```

```
            }

            // 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 FileAnalyzer.cs - контроль цілісності даних

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

namespace RecoveryDisk
{
    /// <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 БЧХ Cloud-технології для NAS (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 БЧХ Cloud-технології для NAS (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)BCHConst.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 БЧХ Cloud-технології для NAS (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 БЧХ Cloud-технології для NAS (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)BCHConst.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 БЧХ Cloud-технології для NAS (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)
                        {
                            //...exitимо 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)
                                {
                                    //...exitимо 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)
                            {
                                //...exitимо 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 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 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.BCHIconTimer = 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 ДИПЛОМНА РОБОТА",
            "",
            "На тему:",
            "",
            "Програмне забезпечення Cloud-технології для NAS",
            "",
            "",
            "Керівник: Петренюк В.І.  ",
            "",
            "Розробив: студент Сеїджанов Ахмед ",
            "                  гр. КІ-20",
            "",
            "м. Кропивницький 2024"});
            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
);
            //
            // BCHIconTimer
            //
            this.BCHIconTimer.Interval = 40;
            this.BCHIconTimer.Tick += new
System.EventHandler(this.BCHIconTimer_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 BCHIconTimer;
        private PinkieControls.ButtonXP okButtonXP;

    }
}
```