

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
« ____ » _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи підвищення надійності
центрів обробки даних Data Center Manager”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Хаджилій К.Р.
« ____ » _____ 2025 р.

Керівник проекту
доктор філософії (PhD)
_____ Усік П.С.
« ____ » _____ 2025 р.
Рецензент _____

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
Олексій СМІРНОВ
« 17 » січня 2025 року

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

Хаджилію Кирилу Романовичу

(прізвище, ім'я, по батькові)

- Тема роботи *Програмне забезпечення системи підвищення надійності центрів обробки даних Data Center Manager*
- Керівник роботи *Усік Павло Сергійович, доктор філософії (PhD)*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу № 46-02 від 17.01.2025 року
- Строк подання студентом роботи до захисту *23.05.2025 р.*
- Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи підвищення надійності центрів обробки даних Data Center Manager*
- Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
 - Призначення та область використання.*
 - Перегляд аналогічних існуючих систем.*
 - Опис і обґрунтування проектних рішень.*
 - Етапи програмування системи.*
 - Впровадження системи в промислову експлуатацію.*
 - Висновки*
- Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

<i>Структурна схема системи</i>	<i>1 аркуш</i>
<i>Функціональна схема системи</i>	<i>1 аркуш</i>
<i>Діаграма процесів</i>	<i>1 аркуш</i>
<i>Блок-схема алгоритму роботи додатку</i>	<i>2 аркуша</i>

7. Дата видачі завдання « 17 » січня 2025 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	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника

Усік П.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача

Хаджилій К.Р.
(прізвище та ініціали)

АНОТАЦІЯ

Хаджилій К.Р. Програмне забезпечення системи підвищення надійності центрів обробки даних Data Center Manager. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи підвищення надійності центрів обробки даних Data Center Manager.

Метою розробки є програмне забезпечення системи підвищення надійності центрів обробки даних Data Center Manager.

Результат роботи – програмна реалізація системи підвищення надійності центрів обробки даних Data Center Manager.

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, Data Center Manager

ABSTRACT

Khadzhiliy K.R. Software for the Data Center Manager data processing center reliability improvement system. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the Data Center Manager data processing center reliability improvement system.

The purpose of the development is the Data Center Manager data processing center reliability improvement software.

The result of the work is the software implementation of the Data Center Manager data processing center reliability improvement system.

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 the software are provided.

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

The program was developed in the Visual C# environment.

Keywords: computer engineering, Data Center Manager

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

КБПЗ-2025

ВСТУП

Актуальність теми. Інженерна інфраструктура ІТ-об'єктів (читай: центрів обробки даних (ЦОД)) – область консервативна. Тому більшість тенденцій, які були відзначені в минулому й навіть позаминулому роках, залишаються актуальними й у році що наступив. Вони пов'язані з підвищенням енергоефективності ЦОД, впровадженням технологій природного охолодження, удосконалюванням засобів керування, а також з поступовим переходом до модульних принципів побудови ЦОД.

Системи керування класу Data Center Manager забезпечують взаємодію інженерної інфраструктури й систем ІТ, і дозволяють, зокрема, здійснювати інтелектуальне виділення ресурсів електроживлення й охолодження. Минулі два роки пішли на своєрідне розгойдування: виробники доводили свої продукти Data Center Manager «до розуму», а потенційні замовники придивлялися до них, зважуючи «за» і «проти» їхнього застосування. Основні вигоди від розгортання Data Center Manager – це зниження числа відмов і продовження строку життя ЦОД, а також скорочення енергоспоживання й економія часу на інвентаризацію.

Крім того, саме інтелектуальні системи керування інженерним устаткуванням, здатні взаємодіяти з ІТ-системами (у тому числі з диспетчерами віртуальних середовищ), дозволять зробити традиційно статичну «інженерію» більше гнучкою, що необхідно для ефективної підтримки динамічних віртуалізованих середовищ.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи підвищення надійності центрів обробки даних Data Center Manager.

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

– Огляд існуючих систем підвищення надійності центрів обробки даних Data Center Manager.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

– Дослідження системи підвищення надійності центрів обробки даних Data Center Manager.

– Програмна реалізація системи підвищення надійності центрів обробки даних Data Center Manager.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі підвищення надійності центрів обробки даних Data Center Manager.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи підвищення надійності центрів обробки даних Data Center Manager, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ – 2025

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

По оцінках експертів Schneider Electric, вартість системи Data Center Manager становить приблизно 1,2% (для корпоративних ЦОД) і 2,2% (для комерційних ЦОД, що надає послуги з розміщення ІТ-устаткування) від витрат на будівництво ЦОД і оснащення його інженерними системами. Це цілком виправдані витрати.

Для роботи будь-якої системи Data Center Manager потрібен збір даних – у першу чергу по енергоспоживанню, температурі й вологості. Звичайно для цієї мети задіюються відповідні датчики, а також «інтелектуальні» блоки розподілу живлення (PDU). Однак у найближчому майбутньому технології, що вбудовуються безпосередньо в ІТ-устаткування, стануть здатні підтримувати функції такого моніторингу. У цьому зв'язку слід зазначити рішення Intel Data Center Manager (Data Center Manager), що забезпечує точний моніторинг споживаної потужності й температури в реальному часі окремих серверів і їхніх груп.

Важливим напрямком зниження енергоспоживання ЦОД, а отже й експлуатаційних витратах, є впровадження технологій природного охолодження (free cooling). Кроком уперед на шляху розширення географії застосування такого охолодження стало відновлення в 2011 році рекомендацій ASHRAE по припустимому рівні робочої температури для ІТ-устаткування. Нагадаю, що в них уведено два нових класи встаткування для ЦОД – А3 і А4, для яких температурний діапазон розширений до +40 і +45°C відповідно. Теоретично поява таких «високотемпературних» ІТ-рішень дозволить мінімізувати використання традиційних кондиціонерів і чиллерів або виключити потребу в них зовсім.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

Але на практиці, принаймні в Україні, нові рекомендації ASHRAE поки не знаходять широкого застосування. У випадку будівництва комерційних ЦОД їхні власники не можуть заздалегідь точно знати, яке встаткування будуть установлювати їхні клієнти, тому інженерна інфраструктура проектується з розрахунком на обслуговування звичайних серверів, для яких температурний поріг становить $+25-27^{\circ}\text{C}$. У корпоративних ЦОД на першому місці стоять питання надійності й відказостійкості, тому інновації в них застосовуються рідко – у тих випадках, коли класичні рішення принципово не підходять. У споруджуваних в Україні ЦОД природне охолодження розглядається лише як доповнення до основних режимів роботи й найчастіше реалізується на рівні чиллерів. Але з'являються й проекти, де режим природного охолодження є домінуючим (Фрикулинг 2.0).

Наприклад, система охолодження на основі роторних регенераторів («теплові колеса») уже більше року перебуває в комерційній експлуатації в ЦОД з потужністю ІТ-навантаження 2,2 Мвт. Вона являє собою два окремих контури: зовнішній, куди подається вуличне повітря, і внутрішній, у якому циркулює повітря ЦОД. Основний елемент системи – роторний регенератор – забезпечує теплообмін між зовнішнім (холодним) повітрям і теплим повітрям, що надходить від ІТ-навантаження. При температурі на вулиці нижче $+22^{\circ}\text{C}$ система повністю переходить на режим природного охолодження, при більше високих температурах додатково підключається чиллерна система. За минулий строк експлуатації коефіцієнт PUE кліматичної системи виявився рівним 1,20. У переважній більшості ЦОД застосовуються класичні статичні ІБЖ. Вони повною мірою дозволяють реалізувати принцип «плати в міру ростів», оскільки можуть підвищуватись малими порціями.

У випадку застосування динамічних систем крок масштабування становить звичайно 1 Мвт. Якщо бізнес-завдання виправдує такий «квант нарощування», то має сенс розглядати цей варіант організації системи безперебійного гарантованого електроживлення.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

Основними тенденціями розвитку статичних інтелектуальних блоків живлення (ІБЖ) є підвищення рівня модульності рішень і підтримка середовищ віртуалізації, наприклад для коректного завершення роботи віртуальних машин або ініціювання їхнього переїзду у випадку погрози відключення електроживлення. Крім того, виробники продовжують розвивати технології підтримки економічних режимів, зокрема шляхом удосконалювання алгоритму перемикання з байпаса на режим подвійного перетворення у випадку погіршення якості вхідної електрики. Однак ці технології вносять істотно менший внесок у справу скорочення енергоспоживання в порівнянні з інноваційними рішеннями по природному охолодженню.

1.2 Область застосування

У цей час основними пріоритетами для адміністраторів центрів обробки даних є підвищення продуктивності, скорочення витрат і підтримка доступності. Колись збереження ресурсів при одночасному збільшенні продуктивності було практично неможливо навіть для найбільш досвідчених організацій в області ІТ і послуг. Так було до появи системи керування інфраструктурою центра обробки даних.

Системи керування ІТ-службами або ІТ-інфраструктурою компанії дозволили значно вдосконалити керування продуктивністю комп'ютерів і дисковим простором. Але це тільки половина історії. Для оптимізації роботи центрів обробки даних необхідно мати можливість кількісного вираження дискового простору, потужності й елементів системи охолодження. На жаль, це, як правило, вимагає більших витрат часу на збір даних вручну або просто сприяє «емпіричним оцінкам», що приводить до додаткового придбання встаткування й підвищує ризики.

Керування потужністю за допомогою Data Center Manager у реальному часі забезпечує візуалізоване подання даних про споживання й доступності

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

ресурсів дискового простору, живлення й охолодження. Data Center Manager починається з визначення робочого діапазону кожного пристрою, зазначеного виробником, що дозволяє точно обчислити підтримуване навантаження. Але Data Center Manager іде набагато далі цієї статичної інформації, забезпечуючи моніторинг і звітність по кожному пристрої в реальному часі. Ця концепція дозволяє здійснювати візуалізацію поточного щоденного енергоспоживання кожного компонента, що подає точну інформацію про витрату й залишок потужності.

Крім того, Data Center Manager підвищує якість планування продуктивності, надає звітність за певний період часу, а також забезпечує моделювання різних аварійних сценаріїв. Тепер ви зможете довідатися, як зміни впливають на споживання ресурсів у часі, і точно прогнозувати доступність у майбутньому.

Такі можливості дозволяють скоротити витрати й ризики. Завдяки Data Center Manager зникає необхідність у надлишкових ресурсах «про всякий випадок», і ви зможете легко ідентифікувати й використовувати ресурси, що простоюють, з метою зниження капіталовкладень, зберігаючи найвищий рівень якості сервісу. Менша кількість устаткування вимагає меншого обсягу ресурсів для монтажу й обслуговування, що дозволяє істотно скоротити поточні витрати.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи підвищення надійності центрів обробки даних Data Center Manager, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		8

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

StruxureWare Data Center Operation

Керування ресурсами й документування роботи ЦОД за допомогою керування запасами.

Оперативний контроль операцій у центрі обробки даних за допомогою керування інфраструктурними ресурсами, калькулятора ефективності використання електроенергії (ЕВЕ), відображення інформаційних сигналів від пристроїв у реальному режимі часу й багаторівневого подання інформації з можливістю деталізації по окремих зонах.

Модуль InfraStruXure Operations підтримує незалежне від виробника пристроїв керування ресурсами інфраструктури з моніторингом стану пристроїв у реальному часі й відображенням даних на схемі фізичного компонування центра обробки даних, а також надає рекомендації з рішення проблемних ситуацій. Багаторівневе подання з можливістю деталізації забезпечує структурований огляд окремих зон центра обробки даних – як глобальне, так і локальне подання з деталізацією до окремих пристроїв. Калькулятор ефективності використання електроенергії (ЕВЕ) надає інформацію із щоденного споживання електроенергії. Для оперативних відновлень «на ходу» пропонується пристрій InfraStruXure Mobile, що забезпечує бездротовий доступ до інформації InfraStruXure Operations.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		9

використовуються для прийняття рішень і планування розраховуючи на поточні й майбутні вимоги до ресурсів інфраструктури.

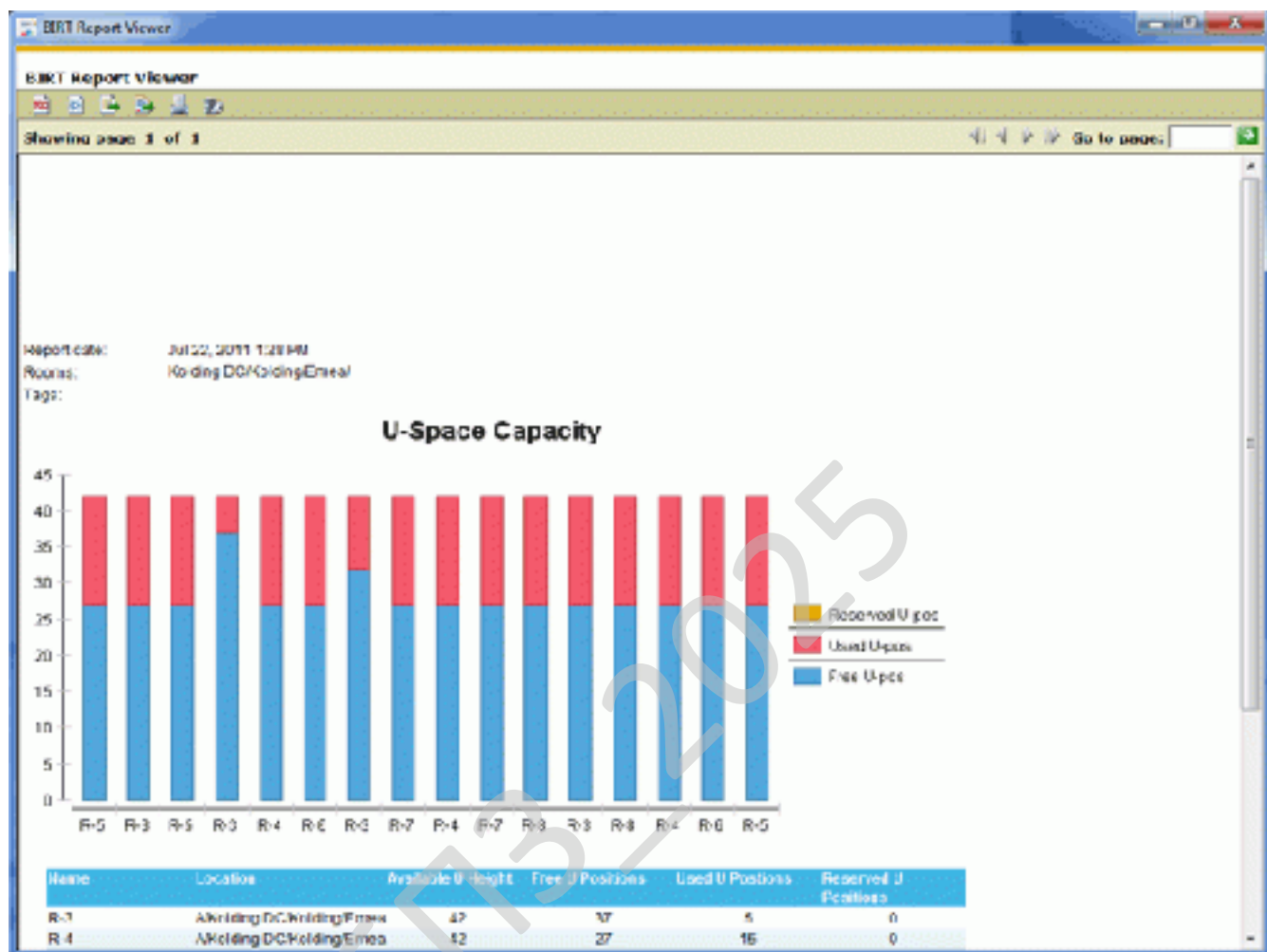


Рисунок 2.2 – Інтерфейс користувача InfraStruXure Capacity

Data Center Operation: IT Optimize

Одержання інформації про енергоспоживання ІТ-систем і їхньому використанні, необхідної для підвищення ефективності й зниження витрат.

Скорочення енергоспоживання ІТ-системи за рахунок глибокої оптимізації використання серверів з погляду підвищення характеристик ЦОДа

StruxureWare Data Center Operation: компонент IT Optimize забезпечує підвищення відсотка використання інфраструктури й ІТ-активів за рахунок

складання точних профілів енергоспоживання ЦОДів з деталізацією до рівня окремої стійки й окремого сервера.

Data Center Operation: Change

Повністю інтегрована система документообігу для інженерної інфраструктури ІТ. Функція обліку наявного встаткування дозволяє з легкістю відслідковувати й виконувати переміщення, установку й модернізацію встаткування в центрі обробки даних.

Модуль InfraStruXure Change дозволяє операторам одержати повний контроль над інфраструктурою центра обробки даних за рахунок впровадження організованої системи для переміщення, додавання й модернізації робочих процесів, що значно знижує ризик позапланового простою. За допомогою автоматизованої системи документообігу оператори можуть формувати вбрання на виконання робіт, резервувати простір, відслідковувати стан і одержувати вибірки з журналу реєстрації подій для повного візуального контролю й ведення історії змін. Мобільний пристрій InfraStruXure Mobile (опція) дозволяє оперативно реєструвати зміни з будь-якого місця центра обробки даних, підтримує сканування штрих-кодів і забезпечує цілісність даних, а також поліпшення експлуатаційної ефективності.

Data Center Operation: Energy Efficiency

Автоматизований аналіз ефективності використання електроенергії (ЕВЕ/шкала DCi) на рівні підсистем.

Повна картина про витрати на споживану електроенергію, а також про поточну і історичну енергоефективність для всього об'єкта, визначення втрат ефективності й поліпшення показника ефективності використання електроенергії (ЕВЕ/шкала DCi) на рівні підсистем.

Модуль InfraStruXure Energy Efficiency надає поточні й історичні значення показника ефективності використання електроенергії (ЕВЕ), що дає обґрунтоване подання про те, яка потужність направляється на роботу встановленого ІТ-устаткування в порівнянні із загальним енергоспоживанням на об'єкті. Дається

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

деталізоване подання про те, наскільки ефективно використовується електроенергія до рівня підсистем, а також надаються рекомендації з поліпшення енергоефективності. Дані про підсистеми можуть вимірятися безпосередньо або оцінюватися за непрямими показниками, що дозволяє замовникам, що не розташовують вимірниками потужності, користуватися всіма перевагами додатка. У веб-формі моніторингу відображаються, зокрема, показники ефективності по поточних і історичних значеннях EUE, а також докладний аналіз вартості споживаної електроенергії на рівні підсистем. Модуль InfraStruXure Energy Efficiency доступний через модуль InfraStruXure Operations, що підтримує інтеграцію із сервером InfraStruXure Central і системами керування корпоративною інфраструктурою сторонніх розроблювачів.

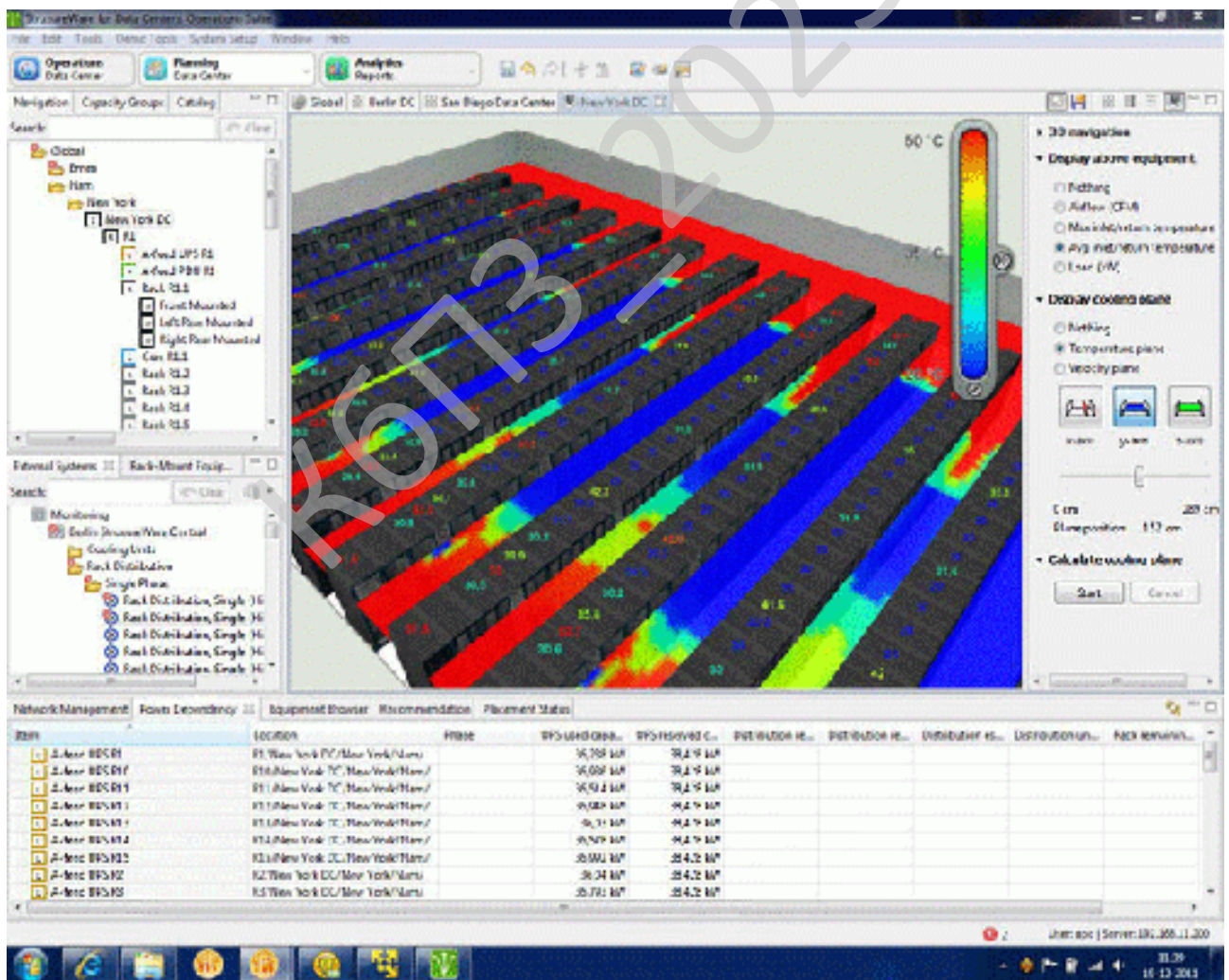


Рисунок 2.3 – Інтерфейс користувача InfraStruXure Energy Efficiency

Data Center Operation: Energy Cost

Оперативна візуалізація енергоспоживання в стійках.

Аналіз вартості споживаної електроенергії в кіловат-годинах з деталізацією до рівня стійки для розрахунку рівня витрат на електроживлення заданого встаткування, об'єктивного виставлення рахунків споживачам і підвищення фінансової ефективності.

Модуль InfraStruXure Energy Cost формує звіт по використанню електроенергії, що показує споживану в центрі обробки даних електроенергію в кіловат-годинах і її вартість по поточному тарифі, з деталізацією на рівні стійки. Показники енергоспоживання засновані на обмірюваних даних, зібраних протягом заданого періоду часу. При відсутності обмірюваних даних очікуване енергоспоживання буде розраховуватися на основі енергоспоживання окремих ІТ-ресурсів або номінальних значень. У звіті по використанню електроенергії (Energy Usage Report) є можливість включення коефіцієнта накладних витрат, що дозволяє врахувати енергетичні втрати при розрахунку показника ефективності використання електроенергії (ЕВЕ). Цей звіт може настрюватися індивідуально з використанням угруповання по окремих категоріях, наприклад, по департаментах, орендарям, призначенню, щільності й т.д.

Data Center Operation: VIZOR

Високорівневі ключові параметри центра обробки даних доступні «на ходу» через планшетний комп'ютер або смартфон.

Перегляд ключових параметрів ЦОДа: характеристики електроживлення, охолодження, робітника простору й локальної мережі, а також високорівневі показники використання ресурсів ЦОДа, які направляються на ваш смартфон або планшетний комп'ютер

Додаток StruxureWare Operations: VIZOR надає ключові параметри ЦОДа: характеристики електроживлення, охолодження, робітника простору й локальної мережі, а також прогнози резерву потужності з урахуванням поточних темпів росту, які направляються прямо на ваш смартфон або планшетний комп'ютер.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		14

Завдяки його простому інтерфейсу користувача радикально спрощується завдання спостереження за станом вашого центра обробки даних «на ходу» – за допомогою функції деталізації ситуації в будь-якій зоні або кімнаті, що забезпечує повне візуальне подання про стан всіх ресурсів центра обробки даних. Для спрощення ідентифікації ресурсів можна сканувати штрих-коди за допомогою убудованої камери. Додаток StruxureWare Operations: VIZOR здійснює обмін даними із серверами StruxureWare Operations і є одним зі значимих компонентів програмного комплексу Schneider Electric Data Center Manager (Data Center Manager), даючи мобільному користувачеві повне візуальне подання про використання ресурсів ЦОДа. Модуль StruxureWare Operations: У цей час додаток VIZOR доступно для пристроїв Apple iPhone і iPad, а також смартфонів на платформі Android.

Data Center Operation: Mobile

Бездротове керування центром обробки даних.

Кишеньковий комп'ютер зі сканером штрих-кодів і підтримкою бездротової мережі для перегляду, створення й оперативної синхронізації змін «на ходу». Пристрій побудований на базі КПК Motorola Symbol MC70.

Пристрій InfraStruXure Mobile, побудоване на апаратній платформі Motorola Symbol MC70, дозволить вам працювати з базою пристроїв центра обробки даних, не використовуючи стаціонарний комп'ютер. Інтегрований сканер штрих-кодів полегшує роботу з виконання вбрань на роботі й визначенню встаткування. Використовуючи вашу бездротову мережу, пристрій InfraStruXure Mobile автоматично синхронізує дані про розташування серверів, забезпечуючи цілісність даних, запобігаючи помилки операторів і підвищуючи експлуатаційну ефективність.

Data Center Operation: Insight

Повнофункціональний інструмент індивідуального складання звітів для візуалізації даних. Генератор звітів, легко адаптируємий до індивідуальних потреб, що відслідковує історичні дані по використанню ресурсів ЦОДа й який

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		15

надає повне подання про ключові показники продуктивності ЦОДа. Модуль StruxureWare Operations Модуль Insight – зроблений інструмент розробки звітів, що надає необмежену кількість варіантів індивідуальної адаптації й візуалізації звітів по ЦОДу відповідно до потреб вашого бізнесу. Цей модуль надає шаблони настроювань за замовчуванням, джерела даних, набори даних і параметри фільтра, даючи можливість формувати індивідуальні рішення. Звіти, створені в середовищі StruxureWare Operations: Insight дозволяють поєднувати дані StruxureWare Operations з будь-якими зовнішніми джерелами даних, доступними через веб-служби або бази даних. Будь-який звіт можна опублікувати на сервері StruxureWare Operations або перетворити у файл одного з безлічі доступних форматів.

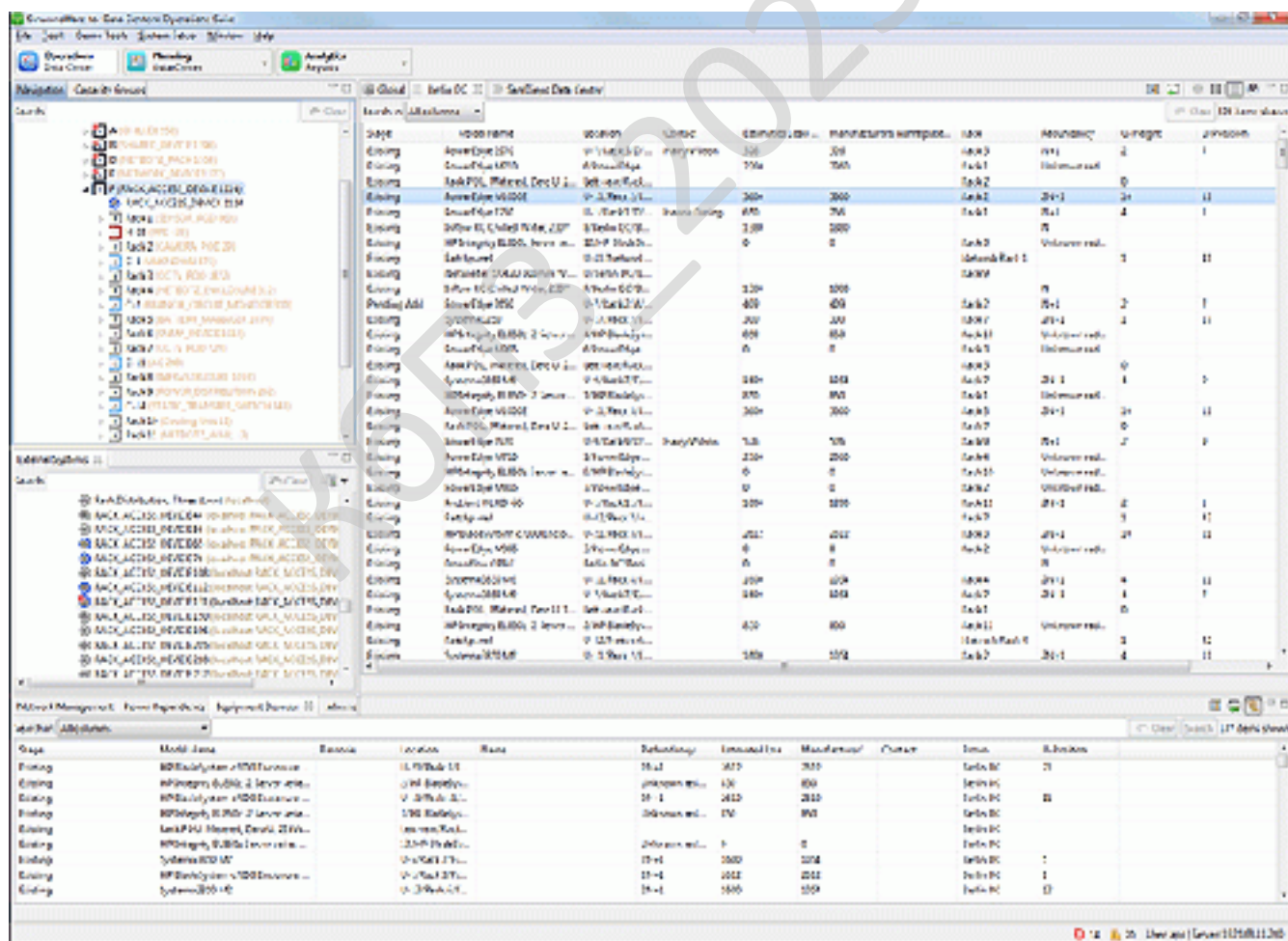


Рисунок 2.4 – Інтерфейс користувача StruxureWare Operations

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

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

- Інкапсульовані підписи методів, називані делегатами, які підтримують строго-типізовані повідомлення про події.
- Властивості, що виступають у ролі методів доступу для закритих змінних-членів.
- Атрибути з декларативними метаданими про типи під час виконання.
- Вбудовані коментарі XML-документації.
- LINQ (Language-Integrated Query), що пропонує вбудовані можливості запитів у різних джерелах даних.

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

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

Архітектура платформи .NET Framework

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

Вихідний код, написаний мовою Visual C#, компілюється в проміжну мову (IL) у відповідності зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення й рядки, зберігаються на диску у файлі, що виконується, названому 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

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		19

на різних мовах платформи .NET Framework, і типи можуть посилатися один на одного, як якби вони були написані на одній мові.

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи підвищення надійності центрів обробки даних Data Center Manager.

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;

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		20

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

КБПЗ - 2025

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		21

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Ефективне використання вільного місця ЦОД стає критичним фактором у продовженні строків його експлуатації. Проблема пов'язана з безперервним ростом енергетичного навантаження: нове обладнання займає багато місця – його закупівлі не завжди виправдані, якщо можна обійтися вертикальною модернізацією. Для рішення завдання потрібні точні розрахунки, засновані на безлічі параметрів.

Необхідна ефективна система, що дозволяє контролювати всі аспекти існування ЦОДа – від параметрів ІТ-устаткування (обчислювальні потужності, системи зберігання даних, мережні параметри) до інфраструктури, що забезпечує (електроустаткування, охолодження, вентиляція, протипожежні системи й системи безпеки) для виходу на оптимальні показники функціонування дата-центра. Системи такого типу, називані Data Center Manager (Datacenter Infrastructure Manager – керування інфраструктурою ЦОД), існують і розвиваються з 2006 року й пропонуються багатьма вендорами встаткування ЦОДів.

Основне завдання Data Center Manager-систем – оптимізація потужностей дата-центра, його охолодження й займаного встаткуванням місця. Збір інформації реалізується інтеграцією до складу Data Center Manager, як більше старих систем керування інфраструктурою будинку (Building Manager System), так і шляхом установки власних датчиків, що контролюють критичні параметри (вологість і температура) навколишнього середовища дата-центра. Data Center Manager-система допомагає поліпшити керування ресурсами, дає можливість створити систему керування інформаційними сервісами, оцінити рівень зрілості процесів

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

керування інформаційними технологіями по методології ITIL (Information Technology Infrastructure Library).

Найбільш розвинені системи містять можливості тестування різних сценаріїв, наприклад при розширенні ЦОДа:

– Чи вистачить потужності ІБЖ при установці додаткових серверних стійок?

– Чи впораються системи охолодження?

Отримана інформація, укупі з варіантами керування, повинна подаватися у двомірному виді й в інтерактивному 3 D-Виді, на одному великому екрані. Всі статистичні показники повинні відображатися на панелях моніторингу – вкладках, кожна з яких відображає різні показники й дозволяє дати оцінку ситуації в дата-центрі, як на рівні окремих обчислювальних вузлів (або інженерного встаткування), так і по ЦОД. Це дозволить операторові бачити що відбувається, оцінювати й оперативню реагувати.

Інтерес до Data Center Manager-систем, відповідно до дослідження Uptime Institute, помітно росте. Причиною тому прискорений ріст "хмар", ринок яких в Україні збільшується випереджальними темпами. Ріст підтримує "бум будівництва дата-центрів", незважаючи на похмурі прогнози щодо динаміки розвитку економіки країни.

Однак, незважаючи на очевидні плюси впровадження Data Center Manager-систем, широкому їхньому поширенню заважає відсутність інформації про методи й практику впровадження. Усе ускладнюється й тим, що процес впровадження реалізується в кілька етапів – це може тривати біля року. Більше того, устаткування може поставлятися різними вендорами – це впливає на загальну інтеграцію й утрудняє уведення в експлуатацію. У сукупності всі ці фактори часто приводять до відмови від впровадження Data Center Manager-системи або до неповної інтеграції з інфраструктурою ЦОД.

Виправданням для впровадження настільки складних систем може служити тільки економія засобів, основою для чого служать розрахунки.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

Наприклад: Data Center Manager-систему EMS-3000 компанія Delta для початку апробувала в дата-центрах своїх офісів у Тайвані й Китаї. Оптимізація роботи систем вентиляції, опалення й кондиціонування повітря, ліфтів і висвітлення допомогла домогтися економії витрати електроенергії в розмірі 50%. В абсолютних цифрах це склало 10 млн кВт у рік.

Ефективність енергоспоживання виражається через значення PUE – коефіцієнті ефективності енергоспоживання, розраховується як відношення загального обсягу електроенергії, спожитої ЦОДом, до електроенергії, спожитої обчислювальним устаткуванням. ЦОДи компанії Delta скоротили цей показник з 2 до 1,5–1,6, у середньому. Це результат, що демонструють найсучасніші дата-центри.

Статистика показує, що незважаючи на значні організаційні труднощі й фінансові витрати в питаннях впровадження Data Center Manager-систем, економія засобів для середніх і великих ЦОД вагома настільки, що дозволяє окупити себе – таким фактором впливу на розмір рахунків за споживану електроенергію бізнес не може зневажити.

3.2 Розробка структурної схеми

Структурна схема системи підвищення надійності центрів обробки даних Data Center Manager за допомогою кодека Ріда-Соломона зображена на рисунку 3.3. З цієї схеми ми бачимо, що над вхідними даними, перед записом у ЦОД, відбуваються перетворення кодеком Ріда-Соломона. Після кодування дані записуються у ЦОД. Після цього інформація зберігається на ЦОДі. Саме за рахунок цього відбувається підвищення надійності центрів обробки даних Data Center Manager. При зчитуванні інформації, розроблене програмне забезпечення декодує інформацію, яка зберігається у ЦОДі, і якщо потрібно, після проведення відповідних перевірок, проводить відновлення втраченої інформації. Якщо таке відновлення неможливе, то програма видає відповідне повідомлення.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		24

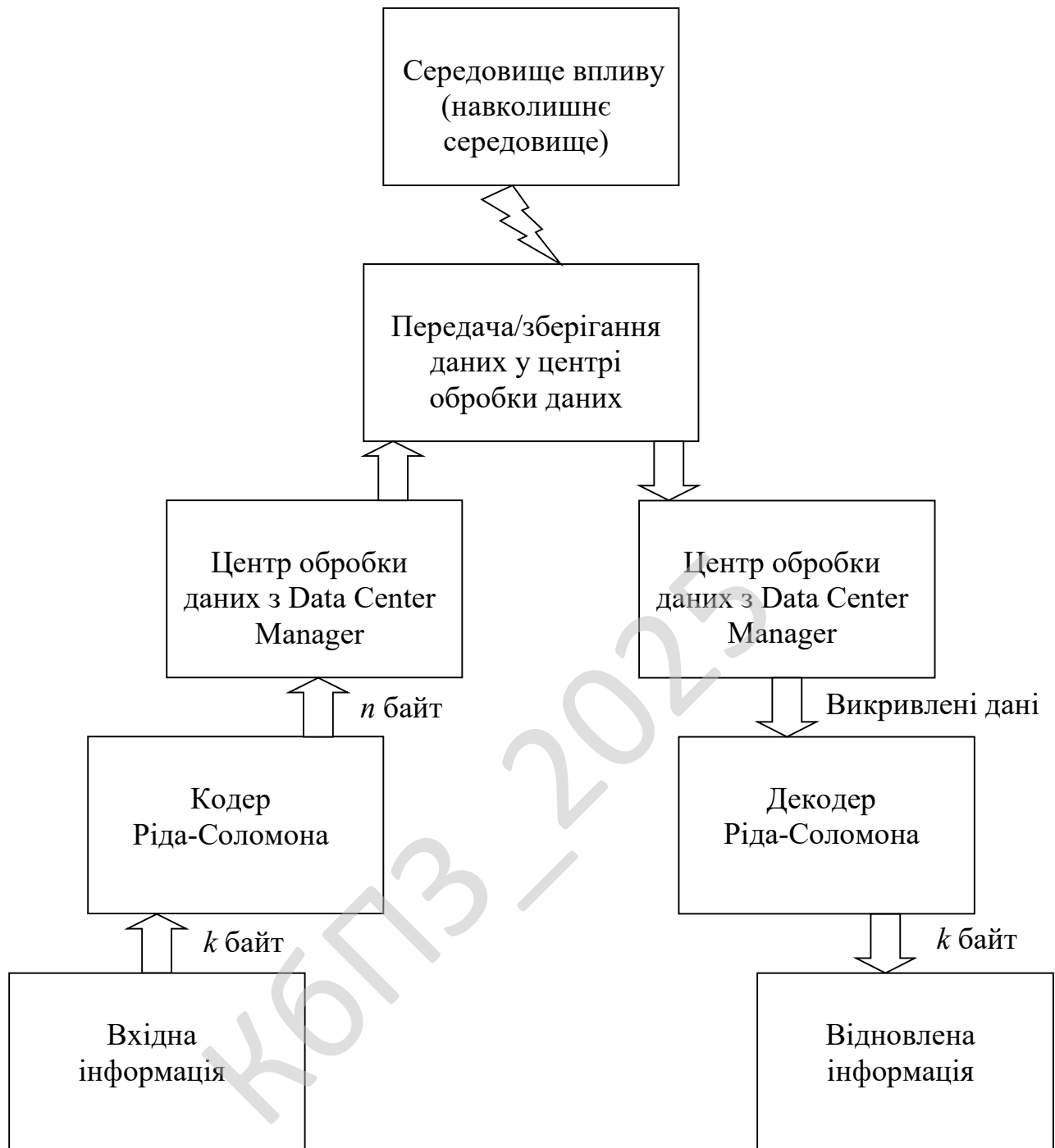


Рисунок 3.1 – Структурна схема системи

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

У цей час широко використовується в системах відновлення даних з компакт-дисків, при створенні архівів з інформацією для відновлення у випадку ушкоджень, у завадостійкому кодуванні.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. З неї бачимо, що розроблена система підвищення стійкості до уражень інформації яка записана у центр обробки даних Data Center Managerах, складається з наступних основних функціональних блоків:

- сам ЦОД;
- кодер Ріда-Соломона, який використовується, коли відбувається запис інформації у центр обробки даних Data Center Manager, при цьому необхідно враховувати, що об'єм інформації, яка записується у центр обробки даних Data Center Manager, повинна бути меншою, ніж об'єм дисків ЦОД, в зв'язку, з тим, що коди Ріда-Соломона відносяться до кодів з надмірністю, за рахунок якої й відбувається кодування;
- декодер Ріда-Соломона, який використовується при читанні даних з відповідного центру обробки даних Data Center Manager.

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

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

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



Рисунок 3.2 – Функціональна схема системи

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

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

Обчислення синдрому

Обчислення синдрому схоже на обчислення парності. Кодове слово Ріда-Соломона має $2t$ **синдромів**, це залежить тільки від помилок (а не переданих кодових слів). Синдроми можуть бути обчислені шляхом підстановки $2t$ коріння утворюючого полінома $g(x)$ в $r(x)$.

Знаходження позицій символічних помилок

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

1. Визначення полінома локації помилок

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

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

Знаходження значень символічних помилок

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

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

3.4 Розробка діаграми процесів

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

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

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

- Зовнішні по відношенню до системи сутності.
- Потоки даних між елементами трьох попередніх типів.



Рисунок 3.3 – Діаграма взаємодії процесів

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

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.

Опис алгоритмів функціонування системи

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об'єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю системи підвищення надійності центрів обробки даних Data Center Manager.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		31

позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

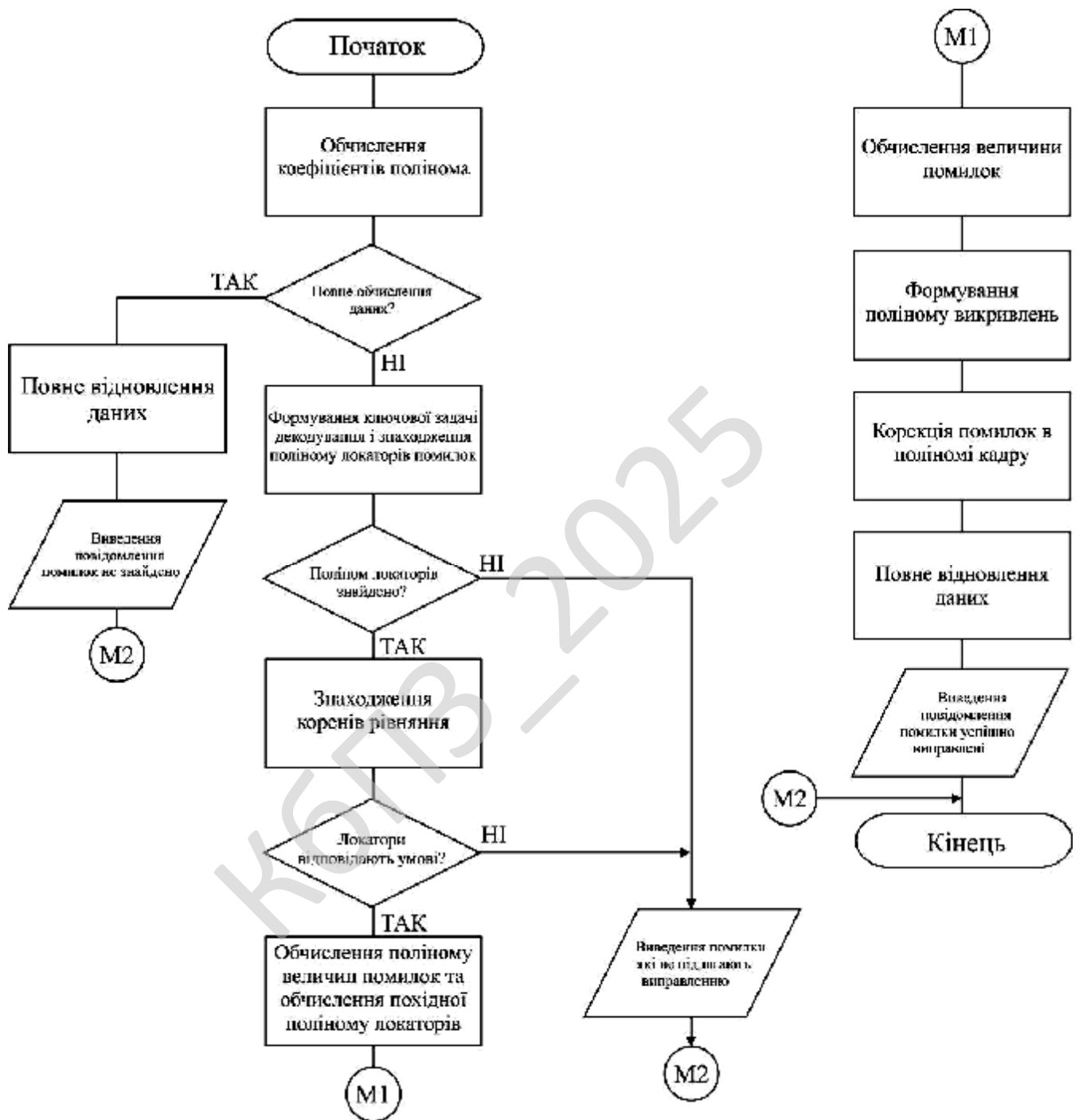


Рисунок 4.2 – Блок-схема роботи підпрограми

діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *).
Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

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

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

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

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

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		35

Декодер Ріда-Соломона

Декодування кодів Ріда-Соломона являє собою досить складне завдання, рішення якого виливається в громіздкий, заплутаний і надзвичайно ненаглядний програмний код, що вимагає від розроблювача великих знань у багатьох областях вищої математики. Типова схема декодування, що одержала назву авторегрессионного спектрального методу декодування, складається з наступних кроків:

- Обчислення синдрому помилки (синдромний декодер).
- Побудови полінома помилки, здійснювана або за допомогою високоефективного, але складно реалізованого алгоритму Берлекемпа-Мессі, або за допомогою простого, але гальмового Евклідового алгоритму.
- Знаходження корінь даного полінома, що звичайно вирішується лобовим перебором (алгоритм Ченя).
- Визначення характеру помилки, що зводиться до побудови бітової маски, що обчислюється на основі обігу алгоритму Форни або будь-якого іншого алгоритму обігу матриці.
- Нарешті, виправлення помилкових символів, шляхом накладення бітової маски на інформаційне слово й послідовне інвертування всіх перекручених біт через операцію XOR.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму FEAL – блоковий шифр, запропонований Акіхіро Симідзу і Седзі Міягуті.

У ньому використовуються 64-бітовий блок і 64-бітовий ключ. Його ідея полягає і в тому, щоб створити алгоритм, подібний DES, але з більш сильною функцією етапу. Використовуючи менше етапів, цей алгоритм міг би працювати швидше. На жаль, дійсність виявилася далекою від цілей проекту.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		39

Як вхід процесу шифрування використовується 64-бітовий блок відкритого тексту. Спочатку блок даних підлягає операції XOR з 64 бітами ключа. Потім блок даних розщеплюється на ліву і праву половини. Об'єднання лівої і правої половин за допомогою XOR утворює нову праву половину. Ліва половина і нова права половина проходять через N етапів (спочатку 4). На кожному етапі половина об'єднується за допомогою функції $F[1]$ з 16 бітами ключа і за допомогою XOR – з лівою половиною, створюючи нову праву половину. Вихідна права половина (на початок етапу) стає новою лівою половиною. Після N етапів (ліва і права половини не переставляти після N-го етапу) ліва половина знову об'єднується з допомогою XOR з правою половиною, утворюючи нову праву половину, потім ліва і права об'єднуються разом в 64-бітове ціле. Блок даних об'єднується за допомогою XOR з іншими 64 бітами ключа і алгоритм завершується.

КБПЗ-2025

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40

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 системи підвищення надійності центрів обробки даних Data Center Manager складається з наступних функціональних блоків:

- Навігаційне меню: Файл; Інструменти; Параметри; Довідка.
- Функції представлені у вигляді графічних кнопок: Кодувати; Декодувати; Перевірити; Виправити.
- Вікна обрання групи обробки.
- Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

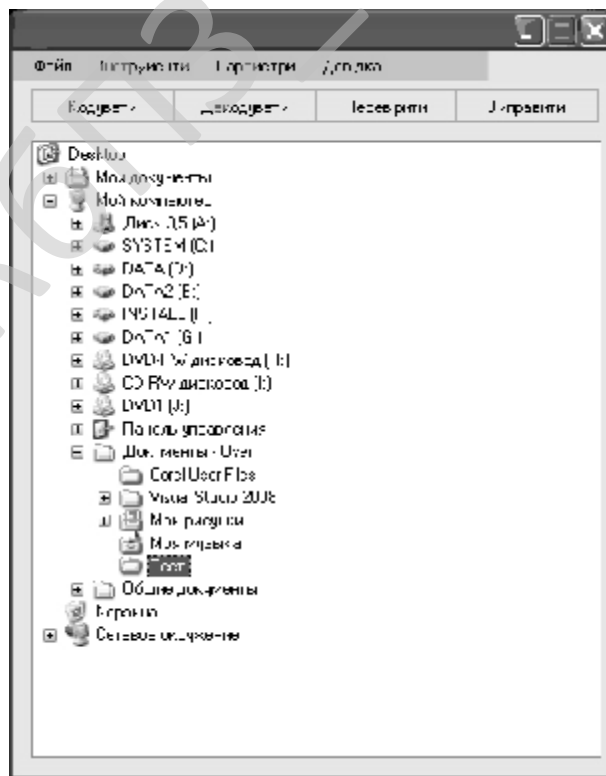


Рисунок 5.1 – Головне вікно розробленого ПЗ

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		41

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.

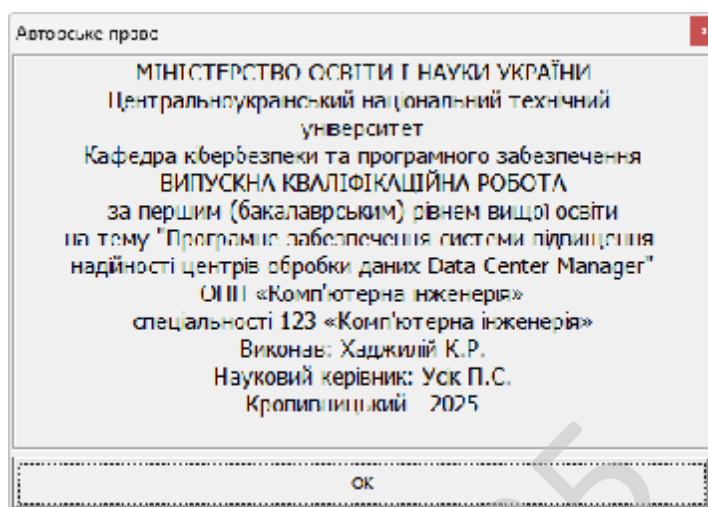


Рисунок 5.2 – Вікно розробника ПЗ

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 білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

- Гарантується перевірка всіх незалежних маршрутів програми.
- Знаходяться гілки True, False для всіх логічних рішень.
- Виконуються всі цикли (у межах їхніх кордонів та діапазонів).
- Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

- Кількість незалежних маршрутів може бути дуже велика.
- Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.
- У програмі можуть бути пропущені деякі маршрути.
- Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

- Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

- Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

- При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

- Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – Shareware.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		43

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ - «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (zareєstrуватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		44

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи підвищення надійності центрів обробки даних Data Center Manager.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем підвищення надійності центрів обробки даних Data Center Manager.

– Досліджена система підвищення надійності центрів обробки даних Data Center Manager.

– На основі отриманих результатів досліджень створена програмна реалізація системи підвищення надійності центрів обробки даних Data Center Manager.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання підвищення надійності центрів обробки даних Data Center Manager.

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

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

Програма реалізована на мові високого рівня Visual C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи підвищення надійності центрів обробки даних Data Center Manager. Це

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		45

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 FEAL.

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.

КБПЗ-2025

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		46

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.
2. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.
3. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.
4. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.
5. Ramon Nastase «Computer Networking: The Beginner’s guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.
6. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.
7. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.
8. 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.
9. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchев, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings*, Volume 3530, 2023, pp. 256-265.
10. 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.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		47

11. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». *CEUR Workshop Proceedings Volume 3156*, 2022, Pages 390-399.

12. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

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

14. 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.

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

18. 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.

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

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

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

22. 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.

23. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

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., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 353-358.

27. 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.

28. 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.

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

30. Смірнова Т.В., Коноплицька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка*. 2024. №4(24), С. 6-27.

31. 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.

32. 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.

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		50

33. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління*, № 2(70). 2022. С. 28-37.

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

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

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

37. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

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

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		51

39. 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.

40. Смірнов О.А., Коноплицька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

41. 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.

42. Смірнов О.А., Дреєва Г.М., Дреєв 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.

43. Смірнов О.А., Коноплицька-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.

44. 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).

45. Смірнов О.А., Дреєва 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

46. Дреєва Г.М., Смірнов О.А., Дреєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		52

модельовання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

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

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

49. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

50. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

51. Смірнов О.А., Смірнов С.А., Дідик А.К., Дреєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

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

					ВКРБ-123.25.0021.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		53

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

					ВКРБ-123.25.0021.00.00.ТЗ			
Вим.	Арк.	№ документа	Підпис	Дата				
Розробив	Хаджилій К.Р.				Програмне забезпечення системи підвищення надійності центрів обробки даних Data Center Manager	Літ.	Аркуш	Аркушів
Перевірів	Усік П.С.					Б	1	6
Н. Контр.	Коваленко А.С.				ЦНТУ КІ-21-1			
Затв.	Смірнов О.А.							

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи підвищення надійності центрів обробки даних Data Center Manager.

2 Підстава для розробки

Підставою для розробки служить завдання на випуск кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 46-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи підвищення надійності центрів обробки даних Data Center Manager.

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;

					ВКРБ-123.25.0021.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		2

- 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 підвищення надійності центрів обробки даних Data Center Manager;
- 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.

					ВКРБ-123.25.0021.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		3

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#.

					ВКРБ-123.25.0021.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		2

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 – 53 аркуші.

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).

					ВКРБ-123.25.0021.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		5

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 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 2.06.2025 р.

					ВКРБ-123.25.0021.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		6

Додаток Б
(обов'язковий)

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за
першим (бакалаврським) рівнем вищої освіти

_____ Усік П.С.

*Програмне забезпечення системи підвищення надійності центрів обробки
даних Data Center Manager*

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 61

Літера: РП

Кропивницький – 2025 року

Файл RSRaidEncoder.cs - кодування алгоритмом підвищення надійності центрів обробки даних Data Center Manager

```

using System;
using System.Threading;

namespace RecoveryDisk
{
    /// <summary>
    /// Клас RAID-подібного кодера підвищення надійності центрів обробки даних
    Data Center Manager
    /// </summary>
    public class RSRaidEncoder : RSRaidBase
    {
        #region Construction & Destruction

        /// <summary>
        /// Конструктор кодера за замовчуванням
        /// </summary>
        public RSRaidEncoder()
        {
            // 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 RSRaidEncoder(int dataCount, int eccCount)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, (int)RSType.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 підвищення надійності центрів
        обробки даних Data Center Manager (по типу матриці)</param>
        public RSRaidEncoder(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 кодера підвищення надійності центрів
обробки даних Data Center Manager (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)RSType.Dispersal)
    {
        maxVolCount = (int)RSConst.MaxVolCountDisp;
    } else
    {
        maxVolCount = (int)RSConst.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.eRSType)
        )
        {
            this.mainConfigChanged = true;
        }

        // Зберігаємо конфігурацію
        this.n = dataCount;
        this.m = eccCount;
        this.eRSType = 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.eRSType == (int)RSType.Alternative)
        {
            this.iterOfFirstStage = m;
        } else
        {
            this.iterOfFirstStage = ((n * m) * n) + (n * ((n + m) + (n *
(n + m))));
        }

        this.iterOfSecondStage = 0; // У кодері 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
        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.eRSType == (int)RSType.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();
            }
        }
    }
}

```

стовпець

```

        return;
    }
} else
{
    //...робимо формування альтернативного заповнення матриці
    "А"
    if (!MakeAlternativeMatrix())
    {
        // Указуємо, що кодер сконфігуровано 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.eRSType == (int)RSType.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 сконфігуровано 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 (OnRSMatrixFormingFinish != null)
{
    //...повідомляємо, що екземпляр класу готовий до роботи
    OnRSMatrixFormingFinish();
}

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

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

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

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

#endregion Private Operations
}
}
```

Декодер підвищення надійності центрів обробки даних Data Center Manager

Файл RSRaidDecoder.cs - декодування алгоритмом підвищення надійності центрів обробки даних Data Center Manager

```

using System;
using System.Threading;

namespace RecoveryDisk
{
    /// <summary>
    /// Клас RAID-подібного декодера підвищення надійності центрів обробки даних
    Data Center Manager
    /// </summary>
    public class RSRaidDecoder : RSRaidBase
    {
        #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 RSRaidDecoder()
        {
            // 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 RSRaidDecoder(int dataCount, int eccCount, int[] volList)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, volList, (int)RSType.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 підвищення надійності центрів
        обробки даних Data Center Manager (по типу матриці)</param>
        public RSRaidDecoder(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 кодера підвищення надійності центрів
обробки даних Data Center Manager (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)RSType.Dispersal)
    {
        maxVolCount = (int)RSConst.MaxVolCountDisp;
    } else
    {
        maxVolCount = (int)RSConst.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.eRSType)
        )
        {
            this.mainConfigChanged = true;
        }

        // Зберігаємо конфігурацію
        this.n = dataCount;
        this.m = eccCount;
        this.eRSType = 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.eRSType == (int)RSType.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 тривіальною, робимо обробку
        if (!this.FLogRowIsTrivial[i])
        {
            int mulSum = 0; // Сума добутку рядка матриці на
            int i_n = i * this.n; // Зсув у масиві до елементів i-ой
            for (int j = 0; j < this.n; j++)
            {
                mulSum ^= GF16Exp[this.FLog[i_n + j] + dataEccLog[j]];
            }
        }
    }
}

```

стовпець

рядка

```

        }

        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-ої 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)
    &&
    (OnUpdateRSMatrixFormingProgress != null)
)
{
    //...виводимо дані
    OnUpdateRSMatrixFormingProgress((((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;
}

///

```

```

}

///  

///  

///  

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 есс-томів для "затикання" пробілів, створених  

    // загубленими основними томами  

    int[] ессVolToFix = 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.eRSType == (int)RSType.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
        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++)
{
    // Шукаємося 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-ої 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.eRSType == (int)RSType.Dispersal)
    {
        int bs = DRowIdx * this.n;

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

```

```

MakeDispersal()
    // "автоматично" на попередньому етапі обробки
    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 (OnRSMatrixFormingFinish != null)
{
    //...повідомляємо, що екземпляр класу готовий до роботи
    OnRSMatrixFormingFinish();
}

// 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 набору файлів-томів

```

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 кодека підвищення надійності центрів обробки даних Data Center
    Manager (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 кодека підвищення надійності центрів
обробки даних Data Center Manager (по типу матриці)</param>
/// <param name="runAsSeparateThread">Запустити 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)RSConst.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 підвищення надійності центрів обробки даних
    Data Center Manager (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 кодека підвищення надійності центрів
обробки даних Data Center Manager (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)RSConst.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 підвищення надійності центрів обробки даних
    Data Center Manager (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 ім Булевська властивість, можливо, ще
// "не виявилось"
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);
}

```

```

"executeEvent"
    // У випадку, якщо потрібна постановка на паузу, подію
    // 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
        // 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",
        // 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 коректно (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 eventId = 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
            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
            this.finished = true;

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

            return;
        }

        //...якщо одержали сигнал про завершення обробки
        if (eventIdx == 2)
        {
            //...exitимо із циклу очікування завершення
            break;
        }
    } // while(true)
} else
{
    // Скидаємо прапор коректності результату
    this.processedOK = false;

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

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

    return;
}

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

```

нас прокинутися

змінних-членів

вкладеним алгоритмом...

(цього й чекали в while(true)!)

членів

потоків

```

    }
}

// 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 постанова 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 альтернативних 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
        // CRC-64, інакше думаємо, що все коректно (цілісність тому
        // 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
                    // то користувач хоче, щоб ми поставили обробку
                    if (!ManualResetEvent.WaitAll(this.executeEvent,
0, false))
                    {
                        //...припиняємо роботу контрольованого
                        this.eFileIntegrityCheck.Pause();

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

                        // А коли прокинулися, указуємо, що обробка
                        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
                    // переходимо на нову ітерацію, тому що
                    // перед постановкою на паузу...
                    if (eventIdx == 0)
                    {
                        //...попередньо скинувши подію, що змусила
                        this.wakeUpEvent[0].Reset();
                    }
                }
            }
        }
    }
}

```

на цілісність
беремо
true))
"executeEvent",
на паузу -
0, false))
алгоритму...
повинна тривати
прокинутися -
прокидаємося
нас прокинутися

```

        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
        this.finished = true;

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

        return;
    }

    //...якщо одержали сигнал про завершення обробки
    if (eventIdx == 2)
    {
        //...exitимо із циклу очікування завершення
        break;
    }
} // while(true)

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

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

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

    return;
}

// У зв'язку із закриттям великої кількості файлових
// необхідно дочекатися запису змін, внесених потоком
// кодування в тіло класу. Потік уже не працює, але
// установлене ім Булевська властивість, можливо, ще
// "не виявилось"
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)

```

змінних-членів

вкладеним алгоритмом...

(цього й чекали в while(true)!)

членів

потоків

```

        {
            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)
{
    //...вказуємо 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
}
}
```

Інтерфейс користувача

Файл MainForm.cs - головні 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 у 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.ToolStripItem();
        this.instrumentToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.adjustmentToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.encodingfilterToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.quickextractionToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.helpToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        this.separatorToolStripMenuItem = new
System.Windows.Forms.ToolStripItemSeparator();
        this.aboutToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        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.ToolStripItem();
        this.testspeedToolStripMenuItem = new
System.Windows.Forms.ToolStripItem();
        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);
        //
        // quickextractionToolStripMenuItem
        //
        this.quickextractionToolStripMenuItem.Name =
"quickextractionToolStripMenuItem";
        this.quickextractionToolStripMenuItem.Size = new
System.Drawing.Size(216, 22);
        this.quickextractionToolStripMenuItem.Text = "Швидке вилучення
диску";
        this.quickextractionToolStripMenuItem.Click += new
System.EventHandler(this.quickextractionToolStripMenuItem_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.ForeColor =
System.Drawing.SystemColors.ControlText;
    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 кодера (основних
томів: ...; томів для відновлення: ...; обсяг виход" +
        "в: ...)";
    //
    // redundancyGroupBox
    //
    this.redundancyGroupBox.Controls.Add(this.redundancyMacTrackBar);
    this.redundancyGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
    this.redundancyGroupBox.ForeColor =
System.Drawing.SystemColors.ControlText;
    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 = "Надмірність кодування:";
    //
    // 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;

```

```

24);
    this.redundancyMacTrackBar.Location = new System.Drawing.Point(6,
    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.toolTip.SetToolTip(this.redundancyMacTrackBar, "Чим більше
надмірність кодування - том вище відмовостійкість, але більше й обсяг " +
"отриманого набору томів");
    this.redundancyMacTrackBar.TrackerColor =
System.Drawing.Color.RoyalBlue;
    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);
    //
    // allVolCountGroupBox
    //
    this.allVolCountGroupBox.Controls.Add(this.allVolCountMacTrackBar);
    this.allVolCountGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
    this.allVolCountGroupBox.ForeColor =
System.Drawing.SystemColors.ControlText;
    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 = "Загальна кількість томів:";
    //
    // 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 = 24;
    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.toolTip.SetToolTip(this.allVolCountMacTrackBar, "Чим більше
томів - том повільніше обробка й вище відмовостійкість");
        this.allVolCountMacTrackBar.TrackerColor =
System.Drawing.Color.RoyalBlue;
        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 = 14;
        this.allVolCountMacTrackBar.ValueChanged += new
EConTech.Windows.MACUI.ValueChangedHandler(this.allVolCountMacTrackBar_ValueChan
ged);
        this.allVolCountMacTrackBar.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.allVolCountMacTrackBar_MouseUp);
        //
        // toolTip
        //
        this.toolTip.AutomaticDelay = 2000;
        this.toolTip.AutoPopDelay = 20000;
        this.toolTip.InitialDelay = 2000;
        this.toolTip.ReshowDelay = 1000;
        //
        // browser
        //
        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 = "Documents and Settings";
        treeNode2.SelectedImageIndex = 20;
        treeNode2.Text = "Documents and Settings";
        treeNode3.Name = "";
        treeNode3.Text = "";
        treeNode4.ImageIndex = 18;
        treeNode4.Name = "Downloads";
        treeNode4.SelectedImageIndex = 20;
        treeNode4.Text = "Downloads";
        treeNode5.Name = "";
        treeNode5.Text = "";
        treeNode6.ImageIndex = 18;
        treeNode6.Name = "Program Files";
        treeNode6.SelectedImageIndex = 20;
        treeNode6.Text = "Program Files";
        treeNode7.Name = "";
        treeNode7.Text = "";
        treeNode8.ImageIndex = 18;
        treeNode8.Name = "RapidDriver";
        treeNode8.SelectedImageIndex = 20;
        treeNode8.Text = "RapidDriver";
        treeNode9.Name = "";
        treeNode9.Text = "";
        treeNode10.ImageIndex = 18;
        treeNode10.Name = "Server";
        treeNode10.SelectedImageIndex = 20;
        treeNode10.Text = "Server";
        treeNode11.Name = "";
        treeNode11.Text = "";

```

```

treeNode12.ImageIndex = 18;
treeNode12.Name = "WINDOWS";
treeNode12.SelectedImageIndex = 20;
treeNode12.Text = "WINDOWS";
treeNode13.Name = "";
treeNode13.Text = "";
treeNode14.ImageIndex = 18;
treeNode14.Name = "WINDOWS.0";
treeNode14.SelectedImageIndex = 20;
treeNode14.Text = "WINDOWS.0";
treeNode15.ImageIndex = 23;
treeNode15.Name = "SYSTEM2 (C:)";
treeNode15.SelectedImageIndex = 24;
treeNode15.Text = "SYSTEM2 (C:)";
this.browser.SelectedNode = treeNode15;
this.browser.ShowFoldersButton = false;
this.browser.ShowNavigationBar = false;
this.browser.Size = new System.Drawing.Size(962, 432);
this.browser.SplitterDistance = 398;
this.browser.StartupDirectory = FileBrowser.SpecialFolders.Other;
this.browser.StartupDirectoryOther = "C:\\";
this.browser.TabIndex = 0;
//
// repairButton
//
this.repairButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
this.repairButton.Image =
((System.Drawing.Image) (resources.GetObject("repairButton.Image")));
this.repairButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
this.repairButton.Location = new System.Drawing.Point(308, 27);
this.repairButton.Name = "repairButton";
this.repairButton.Size = new System.Drawing.Size(100, 97);
this.repairButton.TabIndex = 3;
this.repairButton.Text = "Виправити";
this.repairButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
this.toolTip.SetToolTip(this.repairButton, "Відновити цілісність
відмовостійкого набору томів");
this.repairButton.UseVisualStyleBackColor = true;
this.repairButton.Click += new
System.EventHandler(this.repairButton_Click);
//
// testButton
//
this.testButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
this.testButton.Image =
((System.Drawing.Image) (resources.GetObject("testButton.Image")));
this.testButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
this.testButton.Location = new System.Drawing.Point(210, 27);
this.testButton.Name = "testButton";
this.testButton.Size = new System.Drawing.Size(100, 97);
this.testButton.TabIndex = 4;
this.testButton.Text = "Перевірити";
this.testButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
this.toolTip.SetToolTip(this.testButton, "Перевірити на наявність
помилки відмовостійкий набір томів");
this.testButton.UseVisualStyleBackColor = true;
this.testButton.Click += new
System.EventHandler(this.testButton_Click);
//
// recoverButton
//
this.recoverButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
this.recoverButton.Image =
((System.Drawing.Image) (resources.GetObject("recoverButton.Image")));

```

```

        this.recoverButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
        this.recoverButton.Location = new System.Drawing.Point(111, 27);
        this.recoverButton.Name = "recoverButton";
        this.recoverButton.Size = new System.Drawing.Size(100, 97);
        this.recoverButton.TabIndex = 2;
        this.recoverButton.Text = "Декодувати";
        this.recoverButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
        this.toolTip.SetToolTip(this.recoverButton, "Розкодувати файли з
набору томів з корекцією помилок");
        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 =
((System.Drawing.Image) (resources.GetObject("protectButton.Image")));
        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 = "Кодувати";
        this.protectButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
        this.toolTip.SetToolTip(this.protectButton, "Закодувати файли у
відмовостійкому форматі в даній директорії");
        this.protectButton.UseVisualStyleBackColor = false;
        this.protectButton.Click += new
System.EventHandler(this.protectButton_Click);
        //
        // exitToolStripMenuItem
        //
        this.exitToolStripMenuItem.Image =
global::RecoveryDisk.Properties.Resources.Exit;
        this.exitToolStripMenuItem.Name = "виходToolStripMenuItem";
        this.exitToolStripMenuItem.Size = new System.Drawing.Size(112, 22);
        this.exitToolStripMenuItem.Text = "Вихід";
        this.exitToolStripMenuItem.Click += new
System.EventHandler(this.exitToolStripMenuItem_Click);
        //
        // testspeedToolStripMenuItem
        //
        this.testspeedToolStripMenuItem.Image =
global::RecoveryDisk.Properties.Resources.StartBenchmark;
        this.testspeedToolStripMenuItem.ImageScaling =
System.Windows.Forms.ToolStripItemImageScaling.None;
        this.testspeedToolStripMenuItem.Name =
"тестбистродействияToolStripMenuItem";
        this.testspeedToolStripMenuItem.Size = new System.Drawing.Size(161,
22);
        this.testspeedToolStripMenuItem.Text = "Тест швидкодії";
        this.testspeedToolStripMenuItem.Click += new
System.EventHandler(this.testspeedToolStripMenuItem_Click);
        //
        // MainForm
        //
        this.AutoScaleDimensions = new System.Drawing.Size(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.repairButton);
        this.Controls.Add(this.testButton);
        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 = "Кодек підвищення надійності центрів обробки даних Data
Center Manager - підвищення надійності зберігання даних на CD/DVD";
        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 fileToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem helpToolStripMenuItem;
private System.Windows.Forms.Button protectButton;
private System.Windows.Forms.Button recoverButton;
private System.Windows.Forms.ToolStripMenuItem exitToolStripMenuItem;
private System.Windows.Forms.ToolStripSeparator
separatorToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem aboutToolStripMenuItem;
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
instrumentToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem
testspeedToolStripMenuItem;
private FileBrowser.Browser browser;
private System.Windows.Forms.ToolStripMenuItem
adjustmentToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem
encodingfilterToolStripMenuItem;
private System.Windows.Forms.ToolStripMenuItem
quickextractionToolStripMenuItem;
    }
}

```

Файл ProcessForm.cs - вікно кодування/декодування

```

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 if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </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 томів 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 томів:";
        //
        // logGroupBox
        //
        this.logGroupBox.Controls.Add(this.logListBox);
        this.logGroupBox.Location = new System.Drawing.Point(12, 80);
        this.logGroupBox.Name = "logGroupBox";
        this.logGroupBox.Size = new System.Drawing.Size(871, 130);
        this.logGroupBox.TabIndex = 0;
        this.logGroupBox.TabStop = false;
        this.logGroupBox.Text = "Лог процесу";
        //
        // logListBox
        //
        this.logListBox.BackColor = System.Drawing.SystemColors.Control;
        this.logListBox.BorderStyle = System.Windows.Forms.BorderStyle.None;
        this.logListBox.FormattingEnabled = true;
        this.logListBox.HorizontalScrollbar = true;
        this.logListBox.Location = new System.Drawing.Point(7, 23);

```

```

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

```

```

        this.errorCountLabel_.AutoSize = true;
        this.errorCountLabel_.Location = new System.Drawing.Point(28, 41);
        this.errorCountLabel_.Name = "errorCountLabel_";
        this.errorCountLabel_.Size = new System.Drawing.Size(35, 13);
        this.errorCountLabel_.TabIndex = 0;
        this.errorCountLabel_.Text = "Error :";
        this.toolTip.SetToolTip(this.errorCountLabel_, "Лічильник некоректно
оброблених файлів");
        //
        // okCountLabel_
        //
        this.okCountLabel_.AutoSize = true;
        this.okCountLabel_.Location = new System.Drawing.Point(28, 20);
        this.okCountLabel_.Name = "okCountLabel_";
        this.okCountLabel_.Size = new System.Drawing.Size(28, 13);
        this.okCountLabel_.TabIndex = 0;
        this.okCountLabel_.Text = "OK :";
        this.toolTip.SetToolTip(this.okCountLabel_, "Лічильник коректно
оброблених файлів");
        //
        // toolTip
        //
        this.toolTip.Delay = 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 = "Обработка файла";
        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;  
    }  
}
```

K6ПЗ_2025

Файл BenchmarkForm.cs - вікно тестування швидкодії алгоритму

```

namespace RecoveryDisk
{
    partial class BenchmarkForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
        disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.coderConfigGroupBox = new System.Windows.Forms.GroupBox();
            this.eccCountLabel = new System.Windows.Forms.Label();
            this.dataCountLabel = new System.Windows.Forms.Label();
            this.eccCountLabel_ = new System.Windows.Forms.Label();
            this.dataCountLabel_ = new System.Windows.Forms.Label();
            this.coderSpeedGroupBox = new System.Windows.Forms.GroupBox();
            this.processedDataCountLabel = new System.Windows.Forms.Label();
            this.processedDataCountLabel_ = new System.Windows.Forms.Label();
            this.timeInTestLabel = new System.Windows.Forms.Label();
            this.timeInTestLabel_ = new System.Windows.Forms.Label();
            this.benchmarkTimer = new
System.Windows.Forms.Timer(this.components);
            this.toolTip = new System.Windows.Forms.ToolTip(this.components);
            this.pauseButtonXP = new PinkieControls.ButtonXP();
            this.closeButtonXP = new PinkieControls.ButtonXP();
            this.closingTimer = new System.Windows.Forms.Timer(this.components);
            this.coderConfigGroupBox.SuspendLayout();
            this.coderSpeedGroupBox.SuspendLayout();
            this.SuspendLayout();
            //
            // coderConfigGroupBox
            //
            this.coderConfigGroupBox.Controls.Add(this.eccCountLabel);
            this.coderConfigGroupBox.Controls.Add(this.dataCountLabel);
            this.coderConfigGroupBox.Controls.Add(this.eccCountLabel_);
            this.coderConfigGroupBox.Controls.Add(this.dataCountLabel_);
            this.coderConfigGroupBox.Location = new System.Drawing.Point(12,
90);

            this.coderConfigGroupBox.Name = "coderConfigGroupBox";
            this.coderConfigGroupBox.Size = new System.Drawing.Size(212, 72);
            this.coderConfigGroupBox.TabIndex = 0;
            this.coderConfigGroupBox.TabStop = false;

```

```

this.coderConfigGroupBox.Text = "Конфігурація кодера";
//
// eccCountLabel
//
this.eccCountLabel.AutoSize = true;
this.eccCountLabel.Location = new System.Drawing.Point(161, 47);
this.eccCountLabel.Name = "eccCountLabel";
this.eccCountLabel.Size = new System.Drawing.Size(37, 13);
this.eccCountLabel.TabIndex = 0;
this.eccCountLabel.Text = "65535";
//
// dataCountLabel
//
this.dataCountLabel.AutoSize = true;
this.dataCountLabel.Location = new System.Drawing.Point(161, 25);
this.dataCountLabel.Name = "dataCountLabel";
this.dataCountLabel.Size = new System.Drawing.Size(37, 13);
this.dataCountLabel.TabIndex = 0;
this.dataCountLabel.Text = "65535";
//
// eccCountLabel_
//
this.eccCountLabel_.AutoSize = true;
this.eccCountLabel_.Location = new System.Drawing.Point(11, 47);
this.eccCountLabel_.Name = "eccCountLabel_";
this.eccCountLabel_.Size = new System.Drawing.Size(125, 13);
this.eccCountLabel_.TabIndex = 0;
this.eccCountLabel_.Text = "Томів для відновлення:";
//
// dataCountLabel_
//
this.dataCountLabel_.AutoSize = true;
this.dataCountLabel_.Location = new System.Drawing.Point(11, 25);
this.dataCountLabel_.Name = "dataCountLabel_";
this.dataCountLabel_.Size = new System.Drawing.Size(89, 13);
this.dataCountLabel_.TabIndex = 0;
this.dataCountLabel_.Text = "Основних томів:";
//
// coderSpeedGroupBox
//
this.coderSpeedGroupBox.Controls.Add(this.processedDataCountLabel);
this.coderSpeedGroupBox.Controls.Add(this.processedDataCountLabel_);
this.coderSpeedGroupBox.Controls.Add(this.timeInTestLabel);
this.coderSpeedGroupBox.Controls.Add(this.timeInTestLabel_);
this.coderSpeedGroupBox.Location = new System.Drawing.Point(12, 9);
this.coderSpeedGroupBox.Name = "coderSpeedGroupBox";
this.coderSpeedGroupBox.Size = new System.Drawing.Size(212, 72);
this.coderSpeedGroupBox.TabIndex = 0;
this.coderSpeedGroupBox.TabStop = false;
this.coderSpeedGroupBox.Text = "Швидкість: - Мбайт/з";
this.coderSpeedGroupBox.Enter += new
System.EventHandler(this.coderSpeedGroupBox_Enter);
//
// processedDataCountLabel
//
this.processedDataCountLabel.AutoSize = true;
this.processedDataCountLabel.Location = new System.Drawing.Point(55,
47);

this.processedDataCountLabel.Name = "processedDataCountLabel";
this.processedDataCountLabel.Size = new System.Drawing.Size(10, 13);
this.processedDataCountLabel.TabIndex = 0;
this.processedDataCountLabel.Text = "-";
//
// processedDataCountLabel_
//
this.processedDataCountLabel_.AutoSize = true;
this.processedDataCountLabel_.Location = new
System.Drawing.Point(11, 47);
this.processedDataCountLabel_.Name = "processedDataCountLabel_";

```

```

13);
        this.processedDataCountLabel_.Size = new System.Drawing.Size(40,
        this.processedDataCountLabel_.TabIndex = 0;
        this.processedDataCountLabel_.Text = "Про\`ем:";
        //
        // timeInTestLabel
        //
        this.timeInTestLabel.AutoSize = true;
        this.timeInTestLabel.Location = new System.Drawing.Point(55, 25);
        this.timeInTestLabel.Name = "timeInTestLabel";
        this.timeInTestLabel.Size = new System.Drawing.Size(10, 13);
        this.timeInTestLabel.TabIndex = 0;
        this.timeInTestLabel.Text = "-";
        //
        // timeInTestLabel_
        //
        this.timeInTestLabel_.AutoSize = true;
        this.timeInTestLabel_.Location = new System.Drawing.Point(11, 25);
        this.timeInTestLabel_.Name = "timeInTestLabel_";
        this.timeInTestLabel_.Size = new System.Drawing.Size(30, 13);
        this.timeInTestLabel_.TabIndex = 0;
        this.timeInTestLabel_.Text = "Година:";
        //
        // benchmarkTimer
        //
        this.benchmarkTimer.Interval = 1000;
        this.benchmarkTimer.Tick += new
System.EventHandler(this.BenchmarkTimer_Tick);
        //
        // toolTip
        //
        this.toolTip.AutomaticDelay = 2000;
        this.toolTip.AutoPopDelay = 20000;
        this.toolTip.InitialDelay = 2000;
        this.toolTip.ReshowDelay = 1000;
        //
        // 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(12, 175);
        this.pauseButtonXP.Name = "pauseButtonXP";
        this.pauseButtonXP.Scheme = PinkieControls.ButtonXP.Schemes.Blue;
        this.pauseButtonXP.Size = new System.Drawing.Size(101, 23);
        this.pauseButtonXP.TabIndex = 0;
        this.pauseButtonXP.Text = "Пауза";
        this.toolTip.SetToolTip(this.pauseButtonXP, "Постановка/зняття
процесу тестування продуктивності з паузи");
        this.pauseButtonXP.Click += new
System.EventHandler(this.pauseButtonXP_Click);
        //
        // closeButtonXP
        //
        this.closeButtonXP.BackColor =
System.Drawing.Color.FromArgb(((int)((byte)(0))), ((int)((byte)(236))),
((int)((byte)(233))), ((int)((byte)(216))));
        this.closeButtonXP.DefaultScheme = true;
        this.closeButtonXP.DialogResult =
System.Windows.Forms.DialogResult.None;
        this.closeButtonXP.Hint = "";
        this.closeButtonXP.Location = new System.Drawing.Point(123, 175);
        this.closeButtonXP.Name = "closeButtonXP";
        this.closeButtonXP.Scheme = PinkieControls.ButtonXP.Schemes.Blue;
        this.closeButtonXP.Size = new System.Drawing.Size(101, 23);

```

```

        this.closeButtonXP.TabIndex = 1;
        this.closeButtonXP.Text = "Закрити";
        this.toolTip.SetToolTip(this.closeButtonXP, "Припинення тестування
продуктивності із закриттям даного вікна");
        this.closeButtonXP.Click += new
System.EventHandler(this.closeButtonXP_Click);
        //
        // closingTimer
        //
        this.closingTimer.Tick += new
System.EventHandler(this.closingTimer_Tick);
        //
        // BenchmarkForm
        //
        this.AutoScaleDimensions = new System.Drawing.Size(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(236, 210);
        this.ControlBox = false;
        this.Controls.Add(this.pauseButtonXP);
        this.Controls.Add(this.closeButtonXP);
        this.Controls.Add(this.coderSpeedGroupBox);
        this.Controls.Add(this.coderConfigGroupBox);
        this.DoubleBuffered = true;
        this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedDialog;
        this.MaximizeBox = false;
        this.MinimizeBox = false;
        this.Name = "BenchmarkForm";
        this.ShowIcon = false;
        this.ShowInTaskbar = false;
        this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterParent;
        this.Text = "Підготовка";
        this.Load += new System.EventHandler(this.BenchmarkForm_Load);
        this.coderConfigGroupBox.ResumeLayout(false);
        this.coderConfigGroupBox.PerformLayout();
        this.coderSpeedGroupBox.ResumeLayout(false);
        this.coderSpeedGroupBox.PerformLayout();
        this.ResumeLayout(false);
    }

#endregion

private System.Windows.Forms.GroupBox coderConfigGroupBox;
private System.Windows.Forms.Label dataCountLabel_;
private System.Windows.Forms.Label eccCountLabel_;
private System.Windows.Forms.Label eccCountLabel;
private System.Windows.Forms.Label dataCountLabel;
private System.Windows.Forms.GroupBox coderSpeedGroupBox;
private System.Windows.Forms.Label timeInTestLabel_;
private System.Windows.Forms.Label timeInTestLabel;
private System.Windows.Forms.Label processedDataCountLabel;
private System.Windows.Forms.Label processedDataCountLabel_;
private System.Windows.Forms.Timer benchmarkTimer;
private PinkieControls.ButtonXP closeButtonXP;
private PinkieControls.ButtonXP pauseButtonXP;
private System.Windows.Forms.ToolTip toolTip;
private System.Windows.Forms.Timer closingTimer;
}
}

```

Файл About.cs - вікно довідки про програму

```

namespace RecoveryDisk
{
    partial class About
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(About));
            this.button1 = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.label2 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();

            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(346, 229);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(92, 23);
            this.button1.TabIndex = 1;
            this.button1.Text = "OK";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.label1.Location = new System.Drawing.Point(11, 12);
            this.label1.Name = "label1";

```

```
this.label1.Size = new System.Drawing.Size(144, 13);
this.label1.TabIndex = 2;
this.label1.Text = "БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА";
//
// pictureBox1
//
this.pictureBox1.Image =
((System.Drawing.Image) (resources.GetObject("pictureBox1.Image")));
this.pictureBox1.Location = new System.Drawing.Point(305, -28);
this.pictureBox1.Name = "pictureBox1";
this.pictureBox1.Size = new System.Drawing.Size(133, 132);
this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.Zoom;
this.pictureBox1.TabIndex = 0;
this.pictureBox1.TabStop = false;
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(12, 45);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(49, 13);
this.label2.TabIndex = 3;
this.label2.Text = "на тему:";
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(12, 79);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(257, 13);
this.label3.TabIndex = 4;
this.label3.Text = "Програмне забезпечення підвищення надійності ";
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(12, 102);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(349, 13);
this.label4.TabIndex = 5;
this.label4.Text = "центрів обробки даних Data Center Manager";
//
// label5
//
this.label5.AutoSize = true;
this.label5.Location = new System.Drawing.Point(16, 140);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(139, 13);
this.label5.TabIndex = 6;
this.label5.Text = "Автор: студент Хаджилій Кирило Романович";
//
// label6
//
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(54, 163);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(63, 13);
this.label6.TabIndex = 7;
this.label6.Text = "гр. КІ-21-1";
//
// label7
//
this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(16, 194);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(128, 13);
this.label7.TabIndex = 8;
this.label7.Text = "Керівник: Усік П.С.";
```

```

//
// label8
//
this.label8.AutoSize = true;
this.label8.Location = new System.Drawing.Point(16, 233);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(104, 13);
this.label8.TabIndex = 9;
this.label8.Text = "М. Кропивницький 2024";
//
// About
//
this.AutoScaleDimensions = new System.Drawing.Size(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(445, 258);
this.Controls.Add(this.label8);
this.Controls.Add(this.label7);
this.Controls.Add(this.label6);
this.Controls.Add(this.label5);
this.Controls.Add(this.label4);
this.Controls.Add(this.label3);
this.Controls.Add(this.label2);
this.Controls.Add(this.label1);
this.Controls.Add(this.button1);
this.Controls.Add(this.pictureBox1);
this.Name = "About";
this.Text = "Про програму...";

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
this.ResumeLayout(false);
this.PerformLayout();

}

#endregion

private System.Windows.Forms.PictureBox pictureBox1;
private System.Windows.Forms.Button button1;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label label8;
}
}

```