

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
« ____ » _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи забезпечення
цілісності даних у хмарних сервісах”

Виконав здобувач вищої освіти
II курсу, групи КН-21М-1,4
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Мицак І.М.
« ____ » _____ 2022 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
« ____ » _____ 2022 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань *12* "Інформаційні технології"
Спеціальність *122* "Комп'ютерні науки"
Освітньо-професійна (освітньо-наукова) програма "Комп'ютерні науки"

ЗАТВЕРДЖУЮ

Завідувач кафедри

д.т.н., проф.

Олексій СМІРНОВ

« 6 » вересня 2022 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Мицаку Іллі Миколайовичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 18-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою розробки є дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах*

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

- | | |
|--|---|
| <i>1. Призначення та область використання.</i> | <i>6. Наукова новизна.</i> |
| <i>2. Перегляд аналогічних існуючих систем.</i> | <i>7. Економічна ефективність розробленої програми.</i> |
| <i>3. Опис і обґрунтування проектних рішень.</i> | <i>8. Заходи з охорони праці та техніки безпеки.</i> |
| <i>4. Етапи програмування системи.</i> | <i>9. Висновки.</i> |
| <i>5. Впровадження системи в промислову експлуатацію</i> | |

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

- | | |
|--|-----------------|
| <i>Наукова новизна</i> | <i>1 аркуш</i> |
| <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> |
| <i>Показники економічної ефективності</i> | <i>1 аркуш</i> |

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника

Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача

Мицак І.М.
(прізвище та ініціали)

АНОТАЦІЯ

Мицак І.М. Дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи забезпечення цілісності даних у хмарних сервісах.

Метою розробки є дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах.

Об'єктом дослідження є процес забезпечення цілісності даних у хмарних сервісах.

Предметом дослідження є методи забезпечення цілісності даних у хмарних сервісах.

Методи дослідження базуються на методах забезпечення цілісності даних, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи забезпечення цілісності даних у хмарних сервісах.

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

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

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

Програму розроблено в середовищі Visual C#.

Ключові слова: комп'ютерні науки, цілісність даних, хмарні сервіси

ABSTRACT

Mytsak I.M. Research and software implementation of a system for ensuring data integrity in cloud services. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the system of ensuring the integrity of data in cloud services.

The purpose of the development is the research and software implementation of a system for ensuring the integrity of data in cloud services.

The object of research is the process of ensuring data integrity in cloud services.

The subject of research is the methods of ensuring data integrity in cloud services.

Research methods are based on data integrity methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the system for ensuring the integrity of data in cloud services.

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

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

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

The program was developed in the Visual C# environment.

Keywords: computer science, data integrity, cloud services

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	3
ВСТУП.....	4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	7
1.1 Призначення системи.....	7
1.2 Область застосування.....	8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	10
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.....	10
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування.....	19
2.3 Розгорнута постановка завдання	22
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	24
3.1 Опис функціонування системи	24
3.2 Розробка структурної схеми.....	34
3.3 Розробка функціональної схеми	35
3.4 Розробка діаграми процесів.....	38
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ.....	40
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	40
4.2 Захист розробленого програмного забезпечення.....	56
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	59
6 НАУКОВА НОВИЗНА	67

						ВКРМ-122.22.0007.00.00.ПЗ		
Вим.	Арк.	№ докум.	Підп.	Дата				
Розроб.	Мицак І.М.				Дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах	Літ.	Аркуш	Аркушів
Перев.	Смірнов С.А.					М	1	110
Н.контр.	Гермак В.С.				ЦНТУ КН-21М-1,4			
Затв.	Смірнов О.А.							

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ.....	68
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.....	68
7.2 Розрахунок трудомісткості розробки програмної продукції.....	70
7.3 Визначення чисельності виконавців і планового фонду зарплати.....	72
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника.....	77
7.5 Визначення собівартості розробки та ціни програмної продукції.....	81
7.6 Визначення об'єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції.....	84
7.7 Визначення експлуатаційних витрат.....	84
7.8 Визначення економічної ефективності програмної продукції.....	86
7.9 Висновок.....	88
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ	89
8.1 Вступ.....	89
8.2 Аналіз умов праці	90
8.3 Розробка заходів з охорони праці	92
8.4 Пожежна безпека.....	95
8.5 Розрахункова частина	97
8.6 Висновки до розділу.....	99
9 ОСНОВНІ ВИСНОВКИ.....	100
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	102

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

ВСТУП

Актуальність теми. Сучасні хмарні сервіси – це ефективний засіб зберігання й передачі даних (у них найбільш низька удільна вартість зберігання інформації), потреби ринку дуже великі, і в продаж найчастіше надходять хмарні сервіси, що не відповідають закладеним у формат стандартам. Виробники допускають таке зниження якості лише тому, що коригувальна здатність, закладена в хмарних сервісах досить велика, і навіть не дуже якісний хмарний сервіс, швидше за все, буде й записуватися й зчитуватися, нехай і на знижених швидкостях.

Але основною проблемою такої “другосортності” є те, що “запас міцності” системи корекції помилок хмарних сервісів при роботі з ними вкрай низький, і будь-яка, навіть незначна подряпина на диску серверу хмарного сервісу може ушкодити дані. Імовірність збою збереженого файлу прямо пропорційна кількості займаних їм секторів, і якщо файл – це архів, що займає весь хмарний сервіс, для надійного архівного зберігання інформації потрібно дуже якісний носій.

Існує цілий ряд методів, які дозволяють вирішити цю задачу. Але посеред цього ряду особливо стоять методи завадостійкого кодування, які дозволяють, навіть при досить сильному ушкодженні тексту відтворити його.

Посеред методів завадостійкого кодування в останній час дуже активно розвиваються методи, які засновані на використанні циклічних кодів. Беззаперечною перевагою цих методів є дуже велика здатність виправляти помилки в ушкодженому тексті. Саме до таких методів й відноситься метод Ріда-Соломона. Вони, у цей час широко використовуються в системах відновлення даних з компакт-дисків, при створенні архівів з інформацією для відновлення у випадку ушкоджень, у завадостійкому кодуванні.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах.

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

- Огляд існуючих систем забезпечення цілісності даних у хмарних сервісах.
- Дослідження системи забезпечення цілісності даних у хмарних сервісах.
- Програмна реалізація системи забезпечення цілісності даних у хмарних сервісах.

Об'єктом дослідження є процес забезпечення цілісності даних у хмарних сервісах.

Предметом дослідження є методи забезпечення цілісності даних у хмарних сервісах.

Методи дослідження базуються на методах забезпечення цілісності даних, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

- Удосконалено метод забезпечення цілісності даних у хмарних сервісах.
- Розроблено вітчизняний продукт забезпечення цілісності даних у хмарних сервісах, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі забезпечення цілісності даних у хмарних сервісах.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для підвищення надійності при зберіганні даних у хмарних сервісах. У випадку ушкодження, дані які зберігаються на них, можливо відновити за допомогою завадостійкого кодування. Внесення надмірності на рівні даних практикується досить давно. Для забезпечення підвищеної надійності зберігання даних, у загальному випадку, використовуються комплекси твердих дискових накопичувачів, що реалізують різні типи RAID архітектур, крім цього для резервного копіювання, архівування, структурованого зберігання й відновлення різних даних у необхідний термін. У ході резервного копіювання (backup) створюються копії останніх версій файлів з метою швидкого відновлення працездатності системи у випадку виникнення аварійної ситуації. Такі копії, нерідко називані резервними, зберігаються на носіях (магнітні стрічки, магнітооптичні хмарні сервіси, DVD RAM) певний термін і потім перезаписуються.

Архівне копіювання (archive) являє собою процес створення копій файлів, призначених для безтермінового або довготермінового зберігання. Носії, на яких зберігаються дані копії, називають архівними (диски серверу хмарного сервісу). Періодичне проведення архівного копіювання дозволяє мати копії декількох різних версій тих самих файлів.

Система структурованого зберігання містить у собі пристрою оперативного доступу, що працюють без втручання оператора (HDD і роботизовані бібліотеки магнітооптичних дисків або стрічкових носіїв). Однак великі сховища (у сотні Тбайт) не можуть обмежитися тільки системами автоматичного зберігання. Навіть у них одиночні накопичувачі або, у випадку роботизованих DVD ROM або DVD RAM бібліотек, цілі блоки з них

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

заміняються операторами вручну.

Використання кодів з виявленням і деяким потенціалом для виправлення помилок набуло широкого застосування в засобах телекомунікації.

Внесення надмірності на рівні додатків використовується набагато рідше. Однак, багато хто, особливо мережні, служби споконвічно підтримують можливість роботи з резервним або взагалі з необмеженим заздалегідь невідомою кількістю альтернативних служб. Введення такої можливості рекомендується при розробці програмного забезпечення, однак, сам процес автоматичного перемикання на альтернативну службу повинен підтверджуватися криптографічним обміном первісної (установної) інформацією.

1.2 Область застосування

Областю застосування розроблювального в ході виконання магістерського проекту програмного забезпечення, є підвищення надійності при зберіганні даних у хмарних сервісах.

Дана мета досягається за допомогою кодів які в стані виявити помилки, створені ушкодженням поверхні хмарні сервіси , і виправити їх.

Виявлення помилок – дія, спрямована на контроль цілісності даних при записі/відтворенні інформації або при її передачі по лініях зв'язку. Виправлення помилок (корекція помилок) – процедура відновлення інформації після читання її із пристрою зберігання або каналу зв'язку. Для виявлення помилок використовують коди виявлення помилок, для виправлення – коригувальні коди (коди, що виправляють помилки, коди з корекцією помилок, завадостійкі коди).

Способи боротьби з помилками

У процесі зберігання даних і передачі інформації з мереж зв'язку, або записі/зчитуванні/зберіганні даних на носіях, неминуче виникають помилки. Контроль цілісності даних і виправлення помилок – важливі задачі на багатьох рівнях роботи з інформацією (зокрема, фізичних, канальних, транспортному

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		8

рівнях моделі OSI). У системах можливі кілька стратегій боротьби з помилками:

– виявлення помилок у блоках даних і автоматичний запит повторної передачі ушкоджених блоків – цей підхід застосовується в основному на каналному й транспортному рівнях;

– виявлення помилок у блоках даних і відкидання ушкоджених блоків – такий підхід іноді застосовується в системах потокового мультимедіа, де важлива затримка передачі й немає часу на повторну передачу;

– виправлення помилок (*forward error correction*) застосовується на фізичному рівні.

Коди виявлення й виправлення помилок

Коригувальні коди – коди, що служать для виявлення або виправлення помилок, що виникають при передачі інформації під впливом перешкод, а також при її зберіганні.

Для цього при записі (передачі) у корисні дані додають спеціальним образом структуровану надлишкову інформацію (контрольне число), а при читанні (прийомі) її використовують для того, щоб виявити або виправити помилки. Природно, що число помилок, яке можна виправити, обмежене й залежить від конкретного застосовуваного коду.

З кодами, що виправляють помилки, тісно зв'язані коди виявлення помилок. На відміну від перших, останні можуть тільки встановити факт наявності помилки в переданих даних, але не виправити її. У дійсності, використовувані коди виявлення помилок належать до тих же класам кодів, що й коди, що виправляють помилки. Фактично, будь-який код, що виправляє помилки, може бути також використаний для виявлення помилок (при цьому він буде здатний виявити більше число помилок, ніж був здатний виправити).

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		9

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Огляд кодів

По способу роботи з даними коди, що виправляють помилки, діляться на блоки, що ділять інформацію на фрагменти постійної довжини й обробляють кожний з них окремо, і згорткові, що працюють з даними як з безперервним потоком.

Блокові коди

Нехай кодуема інформація ділиться на фрагменти довжиною k біт, які перетворюються в кодові слова довжиною n біт. Тоді відповідний блоковий код звичайно позначають (n, k) . При цьому число $R = \frac{k}{n}$ називається швидкістю коду.

Якщо вихідні k біт код залишає незмінними, і додає $n - k$ перевірочних, такий код називається систематичним, інакше несистематичним.

Задати блоковий код можна по-різному, у тому числі таблицею, де кожної сукупності з k інформаційних біт зіставляється n біт кодового слова. Однак, гарний код повинен задовольняти, як мінімум, критеріям, що впливають:

- здатність виправляти як можна більше число помилок;
- як можна менша надмірність;
- простота кодування й декодування.

Неважко бачити, що наведені вимоги суперечать один одному. Саме тому існує велика кількість кодів, кожний з яких придатний для свого кола задач.

Практично всі використовувані коди є лінійними. Це пов'язане з тим, що нелінійні коди значно складніше досліджувати, і для них важко забезпечити прийнятну легкість кодування й декодування.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		10

відстань Хеммінга між ними дорівнює 3. Якщо було передане слово A , і канал вніс помилку в одному біті, вона може бути виправлена, тому що навіть у цьому випадку прийняте слово ближче до кодового слова A , чим до будь-якого іншого, і зокрема до B . Але якщо каналом були внесені помилки у двох бітах (у які A відрізнялося від B) то результат помилкової передачі A виявиться ближче до B , чим A , і декодер прийме рішення що передавалося слово B .

Коди Хеммінга

Коди Хеммінга – найпростіші лінійні коди з мінімальною відстанню, тобто здатні виправити одну помилку. Код Хеммінга може бути представлений у такому виді, що синдром $\vec{s} = \vec{r} H^T$, де \vec{r} – прийнятий вектор, буде дорівнює номеру позиції, у якій відбулася помилка. Ця властивість дозволяє зробити декодування дуже простим.

Загальний метод декодування лінійних кодів

Будь-який код (у тому числі нелінійний) можна декодувати за допомогою звичайної таблиці, де кожному значенню прийнятого слова \vec{r}_i відповідає найбільш імовірно передане слово \vec{u}_i . Однак, даний метод вимагає застосування величезних таблиць уже для кодових слів порівняно невеликої довжини.

Для лінійних кодів цей метод можна істотно спростити. При цьому для кожного прийнятого вектора \vec{r}_i обчислюється синдром $\vec{s}_i = \vec{r}_i H^T$. Оскільки $\vec{r}_i = \vec{v}_i + \vec{e}_i$, де \vec{v}_i – кодове слово, а \vec{e}_i – вектор помилки, те $\vec{s}_i = \vec{e}_i H^T$. Потім за допомогою таблиці по синдрому визначається вектор помилки, за допомогою якого визначається передане кодове слово. При цьому таблиця виходить набагато менше, ніж при використанні попереднього методу.

Лінійні циклічні коди

Незважаючи на те, що декодування лінійних кодів уже значно простіше декодування більшості нелінійних, для більшості кодів цей процес усе ще досить складний. Циклічні коди, крім більш простого декодування, володіють і іншими важливими властивостями. Циклічним кодом є лінійний код, що володіє

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

наступною властивістю: якщо \vec{v} є кодовим словом, то його циклічна перестановка також є кодовим словом.

Слова циклічного коду зручно представляти у вигляді багаточленів. Наприклад, кодове слово $\vec{v} = (v_0, v_1, \dots, v_{n-1})$ представляється у вигляді полінома $v(x) = v_0 + v_1x + \dots + v_{n-1}x^{n-1}$. При цьому циклічне зрушення кодового слова еквівалентний множенню багаточлена на x по модулі $x^n - 1$.

Надалі, якщо не зазначене інше, ми будемо вважати, що циклічний код є двійковим, тобто v_0, v_1, \dots можуть приймати значення 0 або 1.

Поліном, що породжує

Можна показати, що всі кодові слова конкретного циклічного коду кратні певному поліному, що породжує, $g(x)$. Поліном, що породжує, є дільником $x^n - 1$.

За допомогою полінома, що породжує, здійснюється кодування циклічним кодом. Зокрема:

– несистематичне кодування здійснюється шляхом множення кодуемого вектора на $g(x)$: $v(x) = u(x)g(x)$;

– систематичне кодування здійснюється шляхом «дописування» до кодуемого слова остачі від ділення $x^{n-k}u(x)$ на $g(x)$, тобто $v(x) = x^{n-k}u(x) + [x^{n-k}u(x) \bmod g(x)]$.

Коди CRC

Коди CRC (cyclic redundancy check – циклічна надлишкова перевірка) є систематичними кодами, призначеними не для виправлення помилок, а для їхнього виявлення. Вони використовують спосіб систематичного кодування, викладений вище: «контрольна сума» обчислюється шляхом ділення $x^{n-k}u(x)$ на $g(x)$. Через те, що виправлення помилок не потрібно, перевірка правильності передачі може вироблятися точно так само. Таким чином, вид полінома $g(x)$ задає конкретний код CRC.

Коди БЧХ

Коди Боуза-Чоудхури-Хоквінгема (БЧХ) є підкласом двійкових циклічних кодів. Їхня відмітна властивість – можливість побудови коду БЧХ із мінімальною

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		13

відстанню не менше заданої. Це важливо, тому що, загалом кажучи, визначення мінімальної відстані коду є дуже складна задача математично полінома $g(x)$ на множники в поле Галуа.

Коди корекції помилок Ріда-Соломона

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

Переваги й недоліки блокових кодів

Хоча блокові коди, як правило, добре справляються з рідкими, але більшими пачками помилок, їхня ефективність при частих, але невеликих помилках (наприклад, у каналі з АБГШ), менш висока.

Згорткові коди

Згорткові коди, на відміну від блокових, не ділять інформацію на фрагменти й працюють із нею як із суцільним потоком даних.

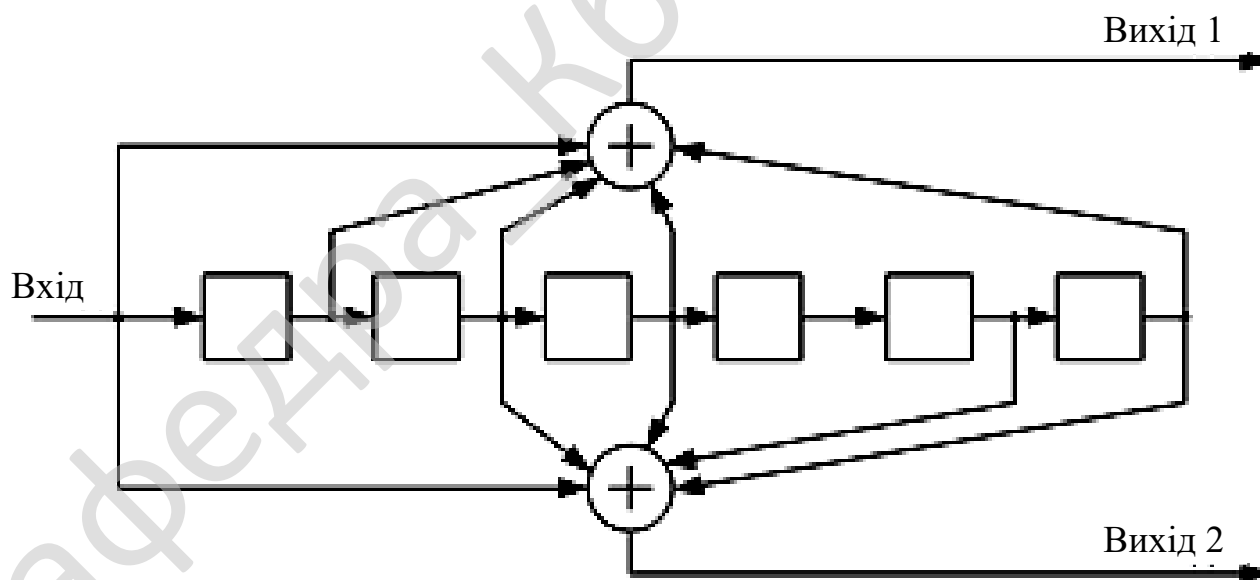


Рисунок 2.1 – Згортковий кодер ($k = 7, R = 1/2$)

Згорткові коди, як правило, породжуються дискретною лінійною інваріантною в часі системою. Тому, на відміну від більшості блокових кодів, згорткове кодування – дуже проста операція, чого не можна сказати про декодування.

Кодування Згортковим кодом відбувається за допомогою реєстра зрушення, відводи від якого підсумуються по модулі два. Таких сум може бути дві (найчастіше) або більше.

Декодування Згорткових кодів, як правило, відбувається по алгоритму Витербі, що намагається відновити передану послідовність відповідно до критерію максимальної правдоподібності.

Переваги й недоліки згорткових кодів

Згорткові коди ефективно працюють у каналі з білим шумом, але погано справляються з пакетами помилок. Більше того, якщо декодер помиляється, на його виході завжди виникає пакет помилок.

Каскадне кодування. Ітеративне декодування

Переваги різних способів кодування можна об'єднати, застосувавши каскадне кодування. При цьому інформація спочатку кодується одним кодом, а потім іншим, у результаті виходить кода-добуток.

Наприклад, популярною є наступна конструкція: дані кодуються кодом Ріда-Соломона, потім перемежуються (при цьому символи, розташовані близько, містяться далеко друг від друга) і кодуються згортковим кодом. На приймачі спочатку декодується згортковий код, потім здійснюється зворотне перемеження (при цьому пачки помилок на виході згорткового декодера попадають у різні кодові слова коду Ріда-Соломона), і потім здійснюється декодування коду Ріда-Соломона.

Деякі коди спеціально сконструйовані для ітеративного декодування, при якому декодування здійснюється в кілька проходів, кожний з яких використовує інформацію від попередні. Це дозволяє домогтися великої ефективності, однак, декодування вимагає більших ресурсів. До таких код відносять турбо-коди й

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		15

LDPC-коди (коди Галлагера).

Оцінка ефективності кодів

Ефективність кодів визначається кількістю помилок, які той може виправити, кількістю надлишкової інформації, додавання якого потрібно, а також складністю реалізації кодування й декодування (як апаратної, так і у вигляді програми для ЕОМ).

Границя Хеммінга й зроблені коди

Нехай є двійковий блоковий (n, k) код з коригувальною здатністю t . Тоді справедлива нерівність (називана границею Хеммінга):

$$\sum_{i=0}^t n_i \leq 2^{n-k} \quad (2.1).$$

Коди, що задовольняють цій границі з рівністю, називаються досконалими. До досконалих кодів відносяться, наприклад, коди Хеммінга. Часто застосовувані на практиці коди з великою коригувальною здатністю (такі, як коди Ріда-Соломона) не є досконалими.

Енергетичний вигреш

При передачі інформації з каналу зв'язку й виробництві запису/читання у хмарних сервісах імовірність помилки залежить від відношення сигнал/шум на вході демодулятора, у такий спосіб при постійному рівні шуму вирішальне значення має потужність передавача. У системах супутникових і мобільної, а також інших типів зв'язку гостро стоїть питання економії енергії. Крім того, у певних системах зв'язку (наприклад, телефонної) необмежено підвищувати потужність сигналу не дають технічні обмеження.

Оскільки завадостійке кодування дозволяє виправляти помилки, при його застосуванні потужність передавача можна знизити, залишаючи швидкість передачі інформації незмінною. Енергетичний вигреш визначається як різниця відносин сигнал/шум при наявності й відсутності кодування.

Застосування кодів, що виправляють помилки

Коди, що виправляють помилки, застосовуються:

– у системах цифрового зв'язку, у тому числі: супутниковій,

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

радіорелейній, стільниковій, передачі даних по телефонних каналах.

– у системах зберігання інформації, у тому числі магнітних і оптичних.

Коди, що виявляють помилки, застосовуються в мережних протоколах різних рівнів.

Автоматичний запит повторної передачі

Системи з автоматичним запитом повторної передачі (ARQ – Automatic Repeat reQuest) засновані на технології виявлення помилок. Поширено наступні методи автоматичного запиту:

– Запит ARQ із зупинками (stop-and-wait ARQ). Ідея цього методу полягає в тім, що передавач очікує від приймача підтвердження успішного прийому попереднього блоку даних перед тим як почати передачу наступні. У випадку, якщо блок даних був прийнятий з помилкою, приймач передає негативне підтвердження (negative acknowledgement, NAK), і передавач повторює передачу блоку. Даний метод підходить для напівдуплексного каналу зв'язку. Його недоліком є низька швидкість через високі накладні витрати на очікування.

– Безперервний запит ARQ з поверненням (continuous ARQ with pullback). Для цього методу необхідний полнодуплексний канал. Передача даних від передавача до приймача відбувається одночасно. У випадку помилки передача відновляється, починаючи з помилкового блоку (тобто, передається помилковий блок і всі наступні).

– Безперервний запит ARQ з вибіркоvim повторенням (continuous ARQ with selective repeat). При цьому підході здійснюється передача тільки помилково прийнятих блоків даних.

Огляд програмного забезпечення

До програмного забезпечення, яке застосовується для підвищення надійності зберігання даних відносяться програмні комплекси, які розглянуті нижче.

Ahead Nero CD Speed

Використовуючи утиліту Ahead Nero CD Speed, протестуйте CD-R/CD-

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		17

RW/DVD-R/DVD-RW хмарні сервіси на предмет виявлення руйнувань.

Тестувати тільки що записані болванки – безглуздо. По-перше, нам необхідно знати характер наростання кількості руйнувань із часом, а, по-друге, – набрати певну статистику надійності по декількох партіях тих самих носіїв.

Для одержання достовірних результатів зовсім необов'язково досліджувати хмарні сервіси записані в MODE 2. Адже з фізичної точки зору режими MODE 1 і MODE 2 зовсім ідентичні один одному, і все, що нам необхідно знати: чи досить здатності CIRC– кодів, що відновлює чи ні.

При тестуванні квадратики, зафарбовані зеленим кольором, означають гарні (good) сектора, помилки читання яких відновлюються ще на рівні CIRC-Декодера. Квадратики, зафарбовані жовтим, символізують частково зруйновані (damaged) сектора, які можливо відновити на рівні MODE 1. На CIRC-рівні такі помилки вже непереборні, і хмарний сервіс, що містить велику кількість damaged-секторів, для запису в режимі MODE 2 категорично непридатний. Червоні квадратики вказують на повністю зруйновані (unreadable) сектора, невідновлювані ні на якому рівні. Присутність навіть єдиних-єдиної-одного-єдиного unreadable сектора сигналізує про ненормальність ситуації й вимагає переходу на більше якісні болванки або ж указує на несправність читаючого/пишучого приводів (наявність руйнувань наприкінці диска цілком припустимо, оскільки тут розташовується 150 секторів області пост-зазору не утримуючих ніяких даних).

Oront burning kit premium

Невеликий, швидкий й простий у використанні додаток для запису дисків формату CD і DVD. Основні функції програми: копіювання, запис, зчитування в образи, очищення дисків, а також підтримка не тільки DVD і CD, але й нових форматів Blu-Ray і HD DVD (звичайно, при наявності відповідного встаткування). Версія Premium, у порівнянні зі стандартною, має можливість вибору файлової системи для носія, режиму запису, редагування томів, створення образів, а також примусового копіювання ушкоджених даних.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		18

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

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

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

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		19

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 серверу хмарного сервісу 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

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		21

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 застосунку 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 другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи забезпечення цілісності даних у хмарних сервісах.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) 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;

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

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.

Кафедра _ КБПЗ _ 2022 рік

					VKPM-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис коду Ріда-Соломона

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

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

Код Ріда-Соломона був винайдений в 1960 році співробітниками лабораторії Лінкольна Массачусетського технологічного інституту Ірвином Рідом і Густавом Соломоном. Ідея використання цього коду була представлена в статті «Polynomial Codes over Certain Finite Fields». Перше застосування код Ріда-Соломона одержав в 1982 році в серійному випуску компакт-дисків. Ефективний алгоритм декодування був запропонований в 1969 році Елвином Берлекемпом і Джеймсом Мессі.

Формальний опис

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

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

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		24

Поліном, що породжує

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

Теорема 3.1

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

Теорема 3.2

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

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

Матриця, що породжує

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

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

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

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

Матрицю G можна записати в символній формі:

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

символів виправляються t помилок (t менш).

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

Виправлення багаторазових помилок

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

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

Практична реалізація

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

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

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

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		27

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

Кодування

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

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

Кодер будується зі регістрів зсуву, суматорів і перемножувачів. Регістр зсуву складається з комірок пам'яті, у кожній з яких перебуває один елемент поля Галуа.

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

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

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

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

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

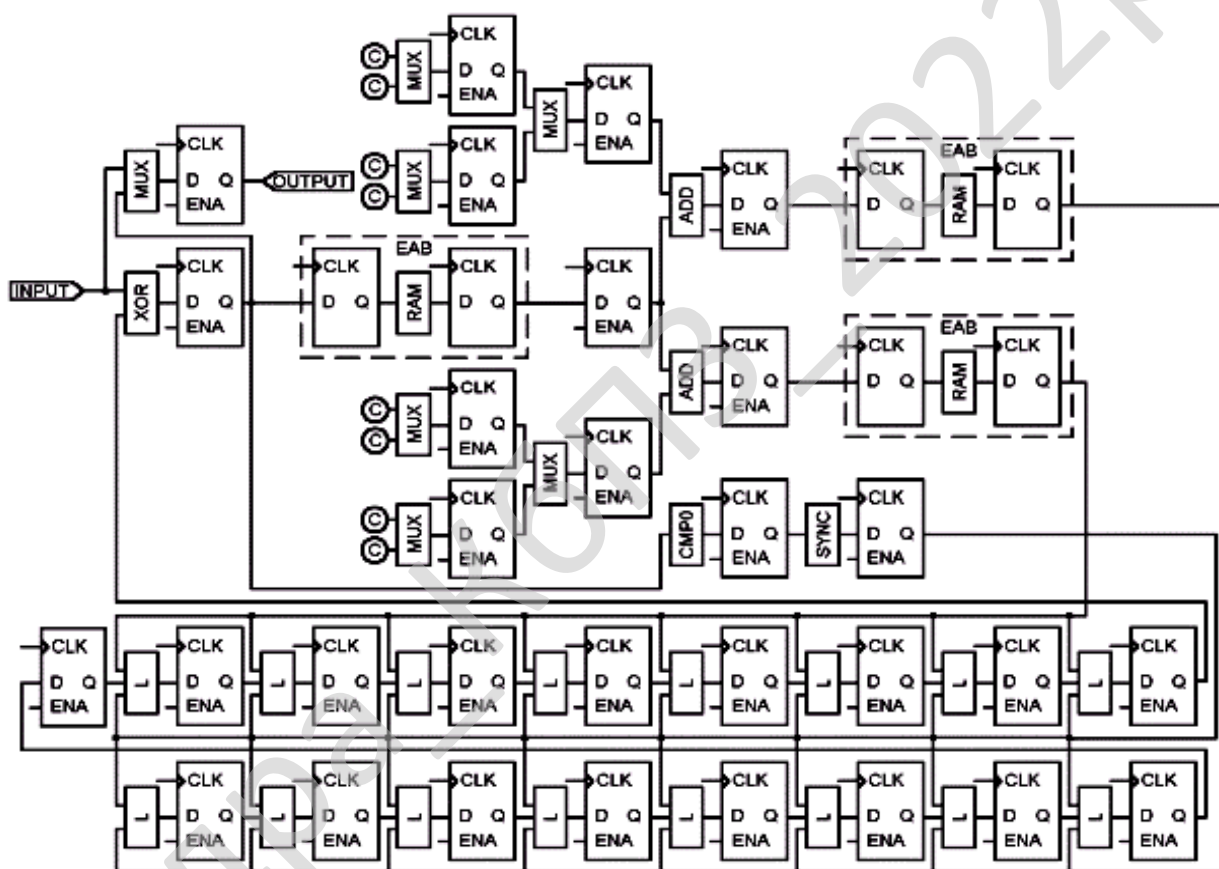


Рисунок 3.2 – Структура кодера Ріда-Соломона

Декодування

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

- Обчислює синдром помилки.

- Будує поліном помилки.
- Знаходить корінь даного полінома.
- Визначає характер помилки.
- виправляє помилки.

Обчислення синдрому помилки

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

Побудова полінома помилки

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

Поліном помилок реалізується за допомогою алгоритму Берлекемпа-Мессі, або за допомогою алгоритму Евкліда. Алгоритм Евкліда має просту реалізацію, але вимагає більших витрат ресурсів. Тому частіше застосовується більше складний, але менш затратоємний алгоритм Берлекемпа-Мессі. Коефіцієнти знайденого полінома безпосередньо відповідають коефіцієнтам помилкових символів у кодовому слові.

Алгоритм Евкліда виконується наступним чином.

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

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

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

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

Коректність цього алгоритму випливає з наступних двох тверджень:

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

Знаходження корня

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

Визначення характеру помилки і її виправлення

По синдрому помилки й знайдених корінь полінома за допомогою алгоритму Форни визначається характер помилки й будується маска перекручених символів. Ця маска накладається на кодове слово за допомогою операції XOR і перекручені символи відновлюються. Після цього відкидаються перевірочні символи й виходить відновлене інформаційне слово.

Застосування

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

Запис і зберігання інформації

Код Ріда-Соломона використовується при записі й читанні в контролерах оперативної пам'яті, при архівуванні даних, запису інформації на жорсткі хмарні сервіси (ЕСС), запису у хмарних сервісах хмарні сервіси. Навіть якщо ушкоджено значний обсяг інформації, зіпсовано кілька секторів дискового носія, то коди Ріда-Соломона дозволяють відновити більшу частину загубленої інформації. Також використовується при записі на такі носії, як магнітні стрічки й

штрихкоди.

Запис у хмарних сервісах-ROM

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

При записі на цифрові аудіокомпакт-хмарні сервіси (Compact Disc Digital Audio – CD-DA) використовується стандарт Red Book. Корекція помилок відбувається на двох рівнях C1 і C2. При кодуванні на першому етапі відбувається додавання перевірочних символів до вихідних даних, на другому етапі інформація знову кодується. Крім кодування здійснюється також перемішування (перемеження) байтів, щоб при корекції блоки помилок розпалися на окремі біти, які легше виправляються. На першому рівні виявляються й виправляються помилкові блоки довжиною один і два байти (один і два помилкових символи відповідно). Помилкові блоки довжиною три байти виявляються й передаються на наступний рівень. На другому рівні виявляються й виправляються помилкові блоки, що виникли в C2, довжиною 1 і 2 байти. Виявлення трьох помилкових символу є фатальною помилкою, не можуть бути виправлені.

Бездротовий і мобільний зв'язок

Цей алгоритм кодування використовується при передачі даних по мережах WiMAX, в оптичних лініях зв'язку, у супутниковому й радіорелейному зв'язку. Метод прямої корекції помилок у минаючому трафіке (Forward Error Correction, FEC) ґрунтується на кодах Ріда-Соломона.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		33

зберігається на носієві.

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

3.3 Розробка функціональної схеми

На рисунку 3.4 зображена функціональна схема системи. З неї бачимо, що розроблена система підвищення стійкості до уражень інформації яка записана на дисках складається з наступних основних функціональних блоків:

- сам носій інформації – хмарний сервіс;
- кодер Ріда-Соломона, який використовується, коли відбувається запис інформації на хмарний сервіс, при цьому необхідно враховувати, що об'єм інформації, яка записується на хмарний сервіс, повинна бути меншою, ніж об'єм диску серверу хмарного сервісу, в зв'язку, з тим, що коди Ріда-Соломона відносяться до кодів з надмірністю, за рахунок якої й відбувається кодування;
- декодер Ріда-Соломона, який використовується при читанні даних с відповідного диску серверу хмарного сервісу.

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

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

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

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		35

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

Декодер працює наступним чином.

Введемо позначення:

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

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

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

Обчислення синдрому

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

3.4 Розробка діаграми процесів

На рисунку 3.5 зображена діаграма взаємодії процесів, які відбуваються у мережі.

Діяльність програми починається з процесу початку/закінчення роботи програми.

Після запуску програми запускаються наступні основні процеси:

- процес кодування;
- процес використання шифруючого фільтра;
- процес декодування;
- процес вибору конфігурації;
- процес тесту швидкості алгоритму.

Розглянемо більш детально взаємодію цих процесів.

Процес кодування взаємодіє з процесами створення томів для відновлення даних та процесом роботи фільтруючого фільтра.

Процес роботи фільтруючого фільтра у свою чергу взаємодіє з процесом встановлення паролю на закодовані дані та процесом декодування.

Процес декодування взаємодіє з наступними процесами:

- процес перевірки цілісності даних;
- процес відновлення пошкоджених томів;
- процес вибору конфігурації.



Рисунок 3.5 – Діаграма процесів системи

Процес вибору конфігурації взаємодіє у свою чергу з такими наступними процесами:

- процес оцінки надмірності кодування;
- процес формування загальної кількості томів.

Розглянувши процеси, які відбуваються в системі перейдемо до опису блок-схем розробленого програмного забезпечення.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображена блок-схема розробленої програми.

З неї бачимо, що програма починає працювати з завантаження головного вікна програми.

У цьому вікні відбувається сканування дисків комп'ютера. З результатами сканування виводиться дерево файлів та папок.

Після цього користувач робить наступні дії:

- вибирає кількість створюваних томів;
- вибирає величину надмірності кодування;
- вибирає папки з файлами для обробки.

Після завдання вищеперахованих дій користувач може виконувати наступні операції.

Якщо натиснута кнопка «Параметри->Шифруючий фільтр» то встановлюється пароль на дані, що кодуються.

Якщо натиснута кнопка «Кодувати» то відбувається виклик підпрограми кодування вибраних файлів алгоритмом Ріда-Соломона, після цього створюються томи для відновлення й відбувається виведення створених томів для відновлення.

Якщо натиснута кнопка «Декодувати» то викликається підпрограма декодування вибраних файлів алгоритмом Ріда-Соломона. Якщо є пошкоджені дані то вони відновлюються за допомогою декодованих томів та корегуючих властивостей кода Ріда-Соломона. Після декодування дані виводяться до файлу.

Якщо натиснута кнопка «Перевірити» то відбувається перевірка цілісності томів для відновлення й виводяться результати перевірки.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40

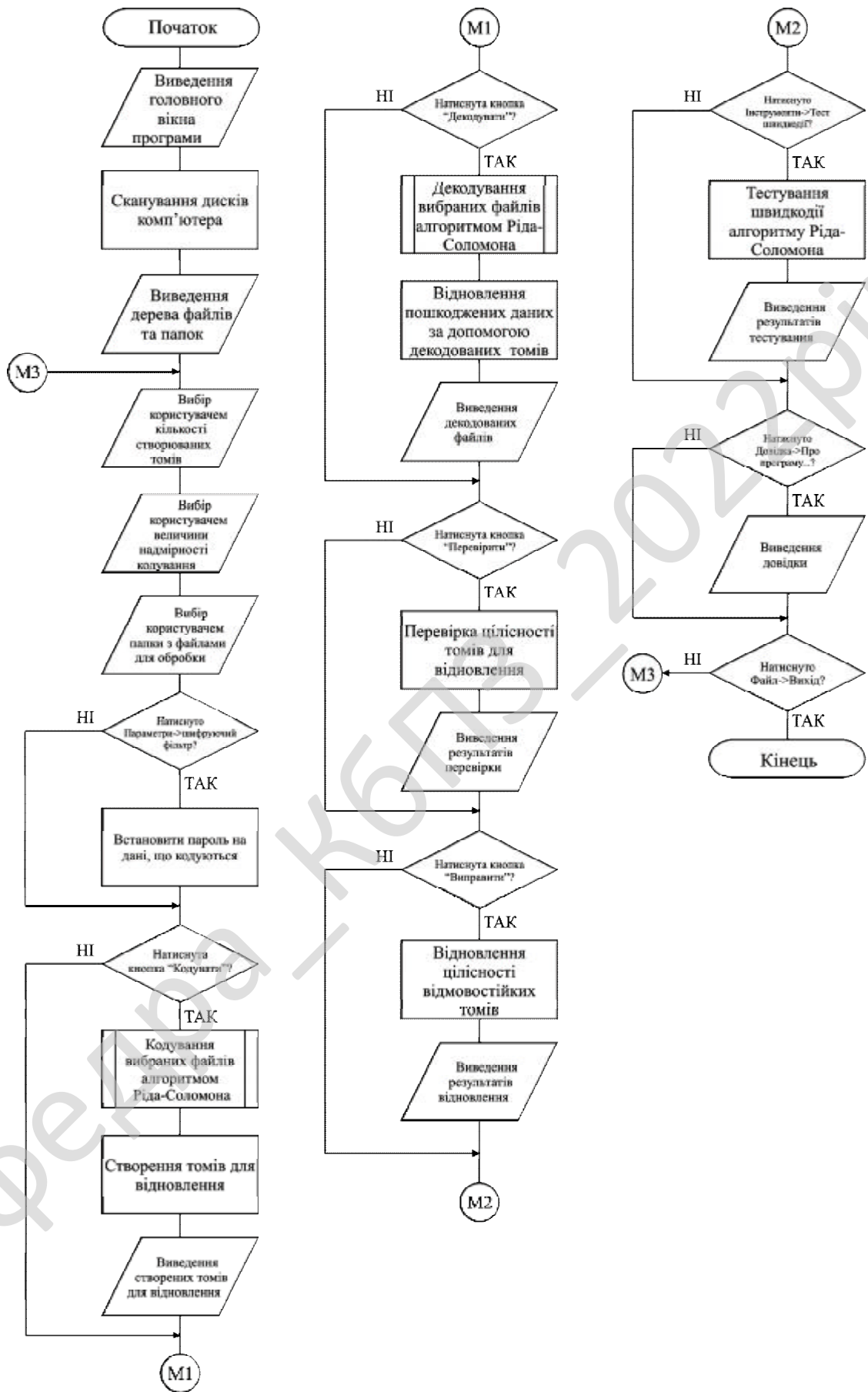


Рисунок 4.1 – Блок-схема роботи основної програми

Якщо натиснута кнопка «Виправити» то відбувається відновлення цілісності відмовостійких томів за рахунок корегуючих властивостей коду Ріда-Соломона.

Якщо натиснута кнопка «Інструмент->Тест швидкодії» то відбувається тестування швидкодії алгоритму Ріда-Соломона. Після тестування користувачу виводяться результати тестування.

Якщо натиснута кнопка «Довідка->Про програму» то виводиться довідка про програму, з вказанням автора програми, назви програми, керівника програми та місця виконання програми.

Якщо натиснута кнопка «Файл->Вихід» то відбувається вихід з програми.

На рисунку 4.3 зображена блок-схема роботи підпрограми кодування алгоритмом Ріда-Соломона.

Перш ніж описувати цю підпрограму наведемо загальні відомості про кодування.

Кодер Ріда-Соломона

Існує, щонайменше, два типи кодерів Ріда-Соломона: несистематичні й систематичні кодувальники.

Обчислення несистематичних коригувальних кодів Ріда-Соломона здійснюється множенням інформаційного слова на породжений поліном, у результаті чого утвориться кодове слово, що повністю відрізняється від вихідного інформаційного слова, а тому для безпосереднього вживання категорично непридатне. Для приведення отриманих даних у вихідний вид ми повинні в обов'язковому порядку виконати ресурсомістку операцію декодування, навіть якщо дані не перекручені й не вимагають відновлення!

При систематичному кодуванні, навпроти, вихідне інформаційне слово залишиться незмінним, а коригувальні коди (часто називані символами парності) додаються в його кінець, завдяки чому до операції декодування доводиться прибгати лише у випадку дійсного руйнування даних. Обчислення несистематичних коригувальних кодів Ріда-Соломона здійснюється розподілом

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		42

інформаційного слова на породжений поліном. При цьому всі символи інформаційного слова зрушуються на $(n - k)$ байт уліво, а на місце, що звільнилося, записується $2t$ байт залишку.

Оскільки розгляд обох типів кодерів зайняло б занадто багато місця, зосередимо свою увагу на одних лише систематичних кодерах, як на найбільш популярні.

Архітектурно кодер являє собою сукупність регістрів зсуву (shift registers), об'єднаних за допомогою суматорів і перемножувачів, що функціонують за правилами арифметики Галуа. Регістр зсуву (інакше називаний регістром зрушення) представляє послідовність комірок пам'яті, названих розрядами, кожний з яких містить один елемент поля Галуа $GF(q)$. Символ, що втримується в розряді, залишаючи цей розряд, "вістрілюється" на вихідну лінію. Одночасно із цим, розряд "засмоктує" символ, що перебуває на його вхідній лінії. Заміщення символів відбувається дискретно, у строго певні проміжки часу, названі тактами.

При апаратній реалізації регістразсуву його елементи можуть бути об'єднані як послідовно, так і паралельно. При послідовному об'єднанні пересилання одного m -розрядного символу зажадає m -тактів, у той час як при паралельному вона здійснюється всього за один такт.

Низька ефективність програмних реалізацій кодерів Ріда-Соломона пояснюється тим, що розроблювач не може здійснювати паралельне об'єднання елементів регістра зсуву й змушений працювати з тією шириною розрядності, що "нав'язує" архітектура даної машини. Однак створити 4-елементний 8-бітний регістр зрушення паралельного типу на процесорах сімейства IA32 цілком реально.

Ланцюги, засновані на регістрах зрушення, звичайно називають фільтрами. Блок-схема фільтра, що здійснює розподіл полінома на константу, наведена на рисунку 4.2. Нехай не бентежить той факт, що розподіл реалізується за допомогою множення й додавання. Даний прийом базується на обчисленні

системи двох рекуррентних рівностей:

$$Q^{(r)}(x) = Q^{(r-1)}(x) + R_{n-r}^{(r-1)} x^{k-r}$$

$$R^{(r)}(x) = R^{(r-1)}(x) - R_{n-r}^{(r-1)} x^{k-r} g(x)$$

Тут: $Q(r)(x)$ і $R(r)(x)$ – відповідно, частка й залишок на r -кроці рекурсії. Оскільки додавання й обчислення, виконуване по модулі два, тотожні один одному, для реалізації дільника нам досить мати всього два пристрої – пристрій додавання й пристрій множення, а без пристрою вирахування можна обійтися.

Після n -зрушень на виході регістра з'являється частка, а в самому регістрі виявиться залишок, що і являє собою розраховані символи парності (вони ж – коди Ріда-Соломона), а коефіцієнти множення з g_0 по $g(2t-1)$ прямо відповідають коефіцієнтам множення породженого полінома.

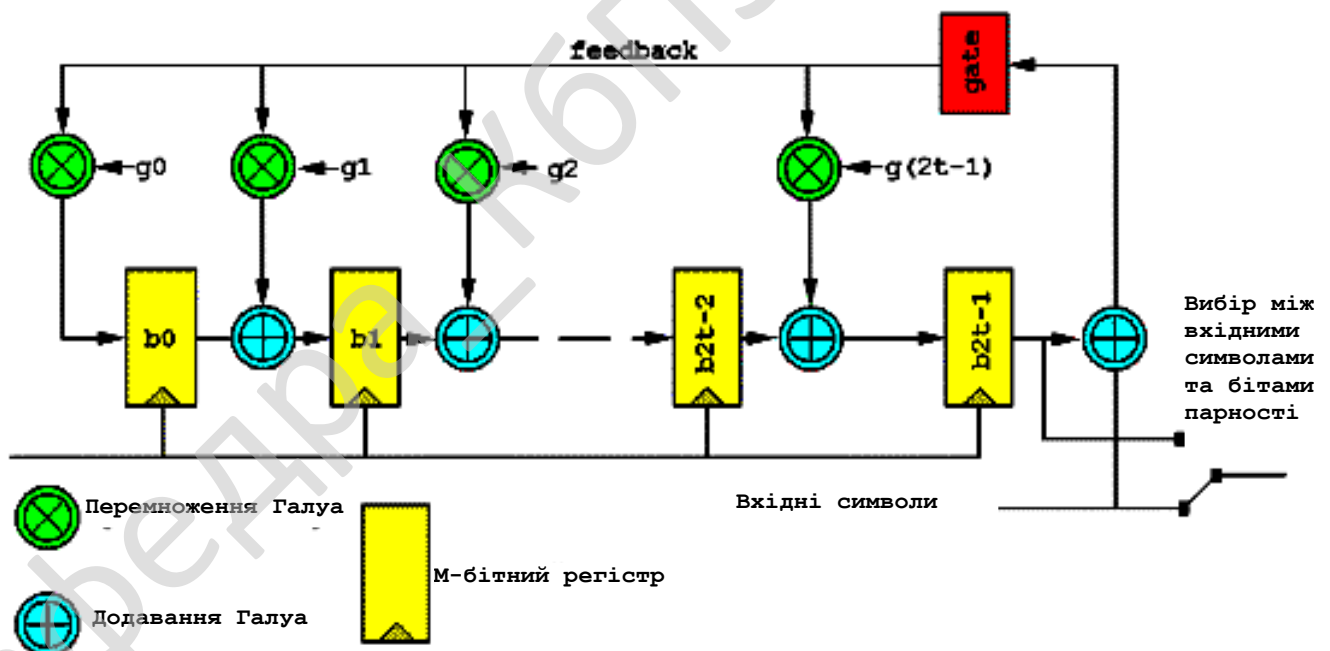


Рисунок 4.2 – Пристрій найпростішого кодера Ріда-Соломона

Найпростіший приклад програмної реалізації такого фільтра наведений нижче. Це закінчений кодера Ріда-Соломона, цілком придатний для практичного

використання. Звичайно, при бажанні його можна було б і поліпшити, але тоді неминуче постраждала б наочність і компактність лістингу.

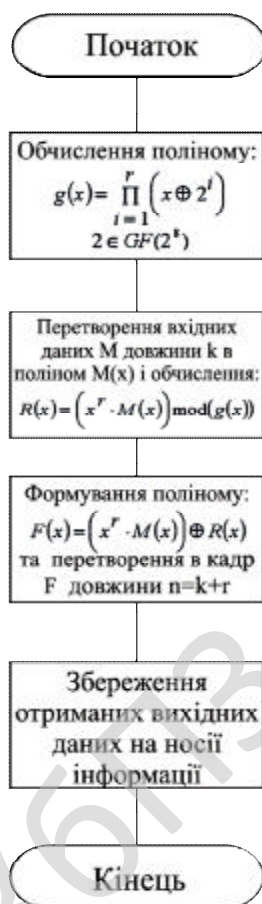


Рисунок 4.2 – Блок-схема роботи підпрограми кодування алгоритмом Ріда-Соломона

Перейдемо до блок схеми. Спочатку відбувається обчислення породжуючого поліному. Після цього вхідні дані перетворюються в поліном для обчислення закодованих даних. Після цього формується поліном $F(x)$ та дані розбиваються покадрово.

В результаті виконання вищеперерахованих дій отримується закодовані дані, які й зберігаються на диску серверу хмарного сервісу.

Нижче наведемо ту частину кода, яка виконує вищеперераховані дії.


```

// і складаємо його з наступних елементів ланцюжка
        if (g[j] != -1) b[j] = b[j - 1] ^ alpha_to[(g[j] +
feedback) % n];
        else
// якщо поточний коефіцієнт g - це нульовий коефіцієнт,
// виконуємо одне лише зрушення без множення, переміщаючи
// символ з одного m-регістра в інший
            b[j] = b[j - 1];
// закріплюємо вихідний символ у крайній лівий b 0-регістр
        b[0] = alpha_to[(g[0] + feedback) % n];
    }
    else
    {
// розподіл завершений,
// здійснюємо останнє зрушення регістра,
// на виході регістра буде частка, що губиться,
// а в самому регістрі - шуканий залишок
        for (j = n - k - 1; j > 0; j-- ) b[j] = b[j - 1]; b[0] = 0;
    }
}
}
}

```

Розглянувши алгоритм та програмну реалізацію процесу кодування перейдемо до процесу декодування закодованого тексту. Блок-схема роботи підпрограми декодування алгоритмом Ріда-Соломона наведена на рисунку 4.4. Але перш ніж перейти до опису підпрограми наведемо короткі теоретичні відомості.

Декодер Ріда-Соломона

Декодування кодів Ріда-Соломона являє собою досить складне завдання, рішення якого виливається в громіздкий, запутаний 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

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		47

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.

Перейдемо до розгляду блок-схеми процедури декодування.

Спершу відбувається перетворення кадру в синдромний поліном та обчислюються коефіцієнти полінома синдрому.

Якщо усі коефіцієнти дорівнюють нулю, то відбувається повне відновлення даних й видається повідомлення, що помилок не знайдено.

У іншому випадку формується ключова задача декодування й знаходження поліному локаторів помилок по алгоритму Берлекемпа-Мессі.

Якщо поліном локаторів не знайдено, то помилки не підлягають виправленню, про що виводиться відповідне повідомлення.

У іншому випадку знаходяться корні рівняння локаторів, виділяються локатори та відбувається перевірка локаторів на умову.

Якщо локатори не відповідають умові то помилки не підлягають виправленню, про що виводиться відповідне повідомлення.

У іншому випадку відбувається обчислення поліному величини помилок, обчислюється його похідна, та обчислюються величини помилок. На основі цих даних формується поліном викривлень та відбувається корекція помилок у кадрі, у результаті чого дані відновлюються, про видається відповідне повідомлення.

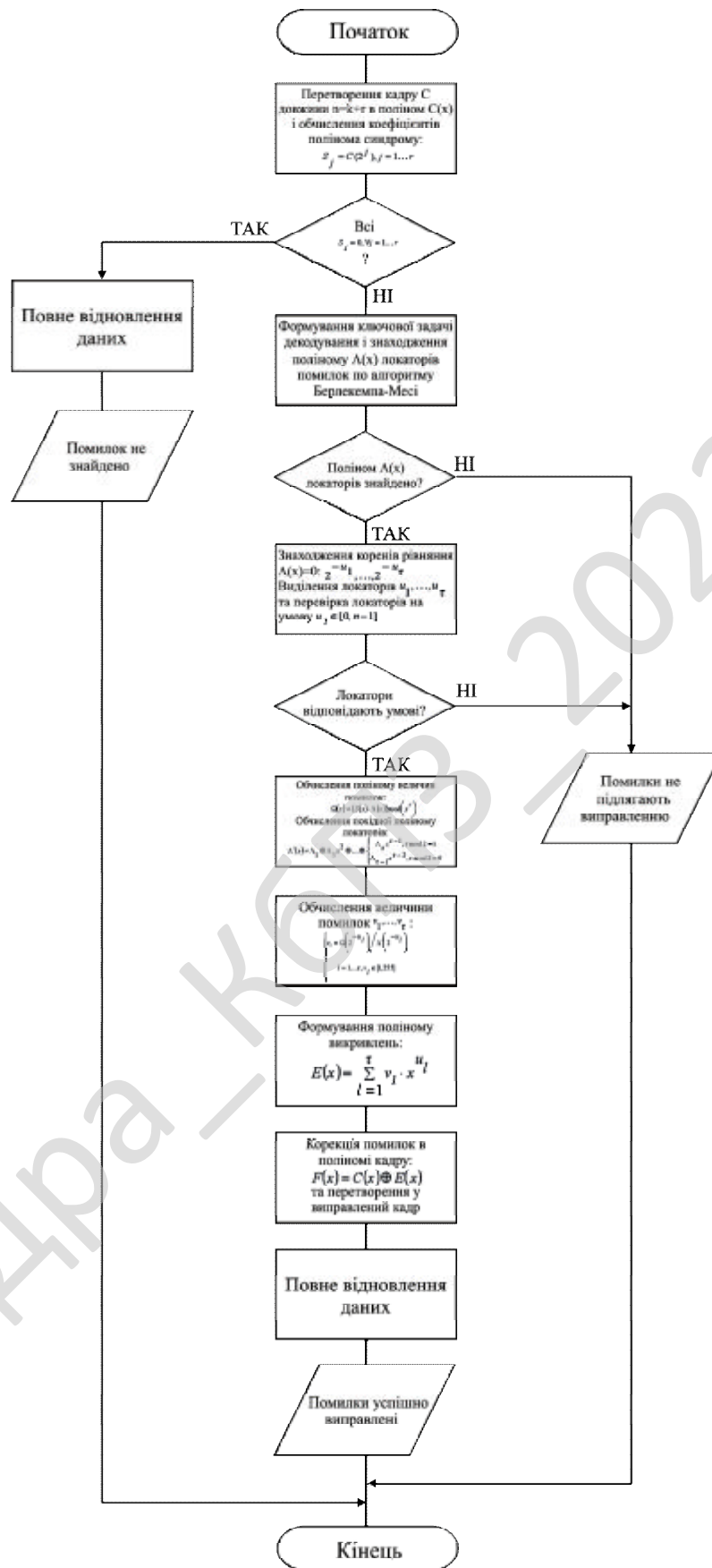


Рисунок 4.4 – Блок-схема роботи підпрограми декодування алгоритмом Ріда-Соломона

Нижче наведемо вихідний текст підпрограми екодера Ріда-Соломона.

```

/*-----
*
* декодер Ріда-Соломона
* =====
*
* Процедура декодування кодів Ріда-Соломона складається з декількох кроків:
* спочатку ми обчислюємо 2 t-символьний синдром шляхом постановки  $\alpha^{*i}$  в
*  $\text{recd}(x)$ , де  $\text{recd}$  - отримане кодове слово, попередньо переведене
* в індексну форму. По факті обчислення  $\text{recd}(x)$  ми записуємо черговий
* символ синдрому в  $s[i]$ , де  $i$  приймає значення від 1 до  $2t$ , залишаючи
*  $s[0]$  рівним нулю.
* потім, використовуючи ітеративний алгоритм Берлекэмп (Berlekamp), ми
* знаходимо поліном локатора помилки -  $\text{elp}[i]$ . Якщо ступінь  $\text{elp}$  перевищує
* собою величину  $t$ , ми неспроможні скорегувати всі помилки й обмежуємося
* виводом повідомлення про непереборну помилку, після чого робимо аварійний
* вихід з декодера. Якщо ж ступінь  $\text{elp}$  не перевищує  $t$ , ми підставляємо
*  $\alpha^{*i}$ , де  $i = 1 \dots n$  в  $\text{elp}$  для обчислення корінь полінома. Обіг
* знайдений корінь дає нам позиції перекручених символів. Якщо кількість
* певних позицій перекручених символів менше ступеня  $\text{elp}$ , перекручуванню
* піддалося більш ніж  $t$  символів і ми не можемо відновити їх.
* У всіх інших випадках відновлення оригінального вмісту
* перекручених символів цілком можливо.
* У випадку, коли кількість помилок свідомо велико, для їхнього виправлення
* декодуємо символи проходять крізь декодер без яких або змін
*
*-----*/
decode_rs()
{
    int i, j, u, q;
    int s[n - k + 1]; // поліном синдрому помилки
    int elp[n - k + 2][n - k]; // поліном локатора помилки лямда
    int d[n - k + 2];
    int l[n - k + 2];
    int u_lu[n - k + 2],
    int count = 0, syn_error = 0, root[t], loc[t], z[t + 1], err[n], reg[t +
1];

    // переводимо отримане кодове слово в індексну форму
    // для спрощення обчислень
    for (i = 0; i < n; i++) recd[i] = index_of[recd[i]];
    // обчислюємо синдром

```

Вим.	Арк.	№ докум.	Підпис	Дата
------	------	----------	--------	------

ВКРМ-122.22.0007.00.00.ПЗ

Арк.

50


```

        elp[0][i] = -1; // індексна форма
        elp[1][i] = 0; // поліноміальна форма
    }
    l[0] = 0; l[1] = 0; u_lu[0] = -1; u_lu[1] = 0; u = 0;
do
{
    u++;
    if (d[u] == -1)
    {
        l[u + 1] = l[u];
        for (i = 0; i <= l[u]; i++)
        {
            elp[u + 1][i] = elp[u][i];
            elp[u][i] = index_of[elp[u][i]];
        }
    }
    else
    {
        // пошук слів з найбільшим u_lu[q], таких що d[q] != 0
        q = u - 1;
        while ((d[q] == -1) && (q > 0)) q--;
        // знайдений перший ненульовий d[q]
        if (q > 0)
        {
            j = q;
            do
            {
                j--;
                if ((d[j] != -1) && (u_lu[q] < u_lu[j]))
                    q = j;
            } while (j > 0);
        };
        // як тільки ми знайдемо q, такий що d[u] !=0
        // i u_lu[q] є максимум
        // запишемо ступінь нового elp полінома
        if (l[u] > l[q] + u - q) l[u + 1] = l[u]; else l[u + 1] =
l[q] + u - q;
        // формуємо новий elp(x)
        for (i = 0; i < n - k; i++) elp[u + 1][i] = 0;
        for (i = 0; i <= l[q]; i++)
            if (elp[q][i] != -1)
                elp[u + 1][i + u - q] = alpha_to[(d[u] + n -

```

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		52

```

d[q] + elp[q][i]) % n];
    for (i = 0; i <= l[u]; i++)
    {
        elp[u + 1][i] ^= elp[u][i];
        // перетворимо старий elp
        // в індексну форму
        elp[u][i] = index_of[elp[u][i]];
    }
}
u_lu[u + 1] = u - l[u + 1];
// формуємо (u + 1)'ю нев'язання
//-----
if (u < n - k) // на останній ітерації розбіжність
{ // не було виявлено
    if (s[u + 1] != -1) d[u + 1] = alpha_to[s[u + 1]]; else d[u +
1] = 0;
    for (i = 1; i <= l[u + 1]; i++)
        if ((s[u + 1 - i] != -1) && (elp[u + 1][i] != 0))
            d[u + 1] ^= alpha_to[(s[u + 1 - i] + index_of[elp[u +
1][i]]) % n];
    // переводимо d[u + 1] в індексну форму
    d[u + 1] = index_of[d[u + 1]];
}
} while ((u < n - k) && (l[u + 1] <= t));
// розрахунок локатора завершений
//-----
u++;
if (l[u] <= t)
{
    // корекція помилок можлива
    // переводимо elp в індексну форму
    for (i = 0; i <= l[u]; i++) elp[u][i] = index_of[elp[u][i]];
// знаходження корінь полінома локатора помилки
//-----
    for (i = 1; i <= l[u]; i++) reg[i] = elp[u][i]; count = 0;
    for (i = 1; i <= n; i++)
    {
        q = 1 ;
        for (j = 1; j <= l[u]; j++)
            if (reg[j] != -1)
            {
                reg[j] = (reg[j] + j) % n;
            }
        }
    }
}

```

```

        q ^= alpha_to[reg[j]];
    }
    if (!q)
    { // записуємо корінь і індекс позиції помилки
        root[count] = i;
        loc[count] = n - i;
        count++;
    }
}
if (count == l[u])
{ // немає корінь - ступінь elp < t помилок
    // формуємо поліном z(x)
    for (i = 1; i <= l[u]; i++) // z[0] завжди дорівнює 1
    {
        if ((s[i] != -1) && (elp[u][i] != -1))
            z[i] = alpha_to[s[i]] ^ alpha_to[elp[u][i]];
        else
            if ((s[i] == -1) && (elp[u][i] == -1))
                z[i] = alpha_to[s[i]];
            else
                if ((s[i] == -1) && (elp[u][i] != -1))
                    z[i] = alpha_to[elp[u][i]];
                else
                    z[i] = 0 ;
        for (j = 1; j < i; j++)
            if ((s[j] != -1) && (elp[u][i - j] != -1))
                z[i] ^= alpha_to[(elp[u][i - j] + s[j]) % n];
        // переводимо z[i] в індексну форму
        z[i] = index_of[z[i]];
    }
}
// обчислення значення помилок у позиціях loc[i]
//-----
for (i = 0; i < n; i++)
{
    err[i] = 0;
    // переводимо recd[] у поліноміальну форму
    if (recd[i] != -1) recd[i] = alpha_to[recd[i]]; else recd[i]
= 0;
}
// спочатку обчислюємо чисельник помилки
for (i = 0; i < l[u]; i++)
{

```

```

err[loc[i]] = 1;
for (j = 1; j <= l[u]; j++)
    if (z[j] != -1)
        err[loc[i]] ^= alpha_to[(z[j] + j * root[i]) %
n];

if (err[loc[i]] != 0)
{
    err[loc[i]] = index_of[err[loc[i]]];
    q = 0 ; // формуємо знаменник коефіцієнта помилки
    for (j = 0; j < l[u]; j++)
        if (j != i)
            q += index_of[1 ^ alpha_to[(loc[j] + root[i])
% n]];

    q = q % n; err[loc[i]] = alpha_to[(err[loc[i]] - q +
n) % n];

    // recd[i] повинен бути в поліноміальній формі
    recd[loc[i]] ^= err[loc[i]];
}
}
}

else // немає корінь,
    // рішення системи рівнянь неможливо, тому що ступінь elp >=
t
{
    // переводимо recd[] у поліноміальну форму
    for (i = 0; i < n; i++)
        if (recd[i] != -1) recd[i] = alpha_to[recd[i]];
    else
        recd[i] = 0; // виводимо інформаційне слово як є
}

else // ступінь elp > t, рішення неможливо
{
    // переводимо recd[] у поліноміальну форму
    for (i = 0; i < n; i++)
        if (recd[i] != -1)
            recd[i] = alpha_to[recd[i]];
        else
            recd[i] = 0 ; // виводимо інформаційне слово як є
}

else // помилок не виявлене
    for (i = 0; i < n; i++) if (recd[i] != -1) recd[i] = alpha_to[recd[i]];
else recd[i] = 0;
}

```

						ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата			55

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму Camellia – блоковий шифр на основі мережі Фейстеля. У криптографії, Camellia – це симетричний ключ блоковий шифр із розміром блоку 128 біт і розмірами ключа 128, 192 і 256 біт. Він був розроблений спільно Mitsubishi Electric і NTT з Японії. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. шифр має рівні безпеки й можливості обробки, порівнянні з Advanced Encryption Standard.

Шифр був розроблений, щоб підходити як для програмних, так і для апаратних реалізацій, від недорогих смарт-карти для високошвидкісних мережних систем. Він є частиною криптографічного протоколу Transport Layer Security (TLS), призначеного для забезпечення безпеки зв'язки в комп'ютерній мережі, такий як Інтернет

Camellia – це шифр Фейстеля з 18 раундами (при використанні 128-бітних ключів) або 24 раундами (при використанні 192- або 256-бітних ключів). Кожні шість раундів застосовується шар логічного перетворення: так звана «FL-функція» або її зворотна. Camellia використовує чотири 8×8 -бітних S-блоку із вхідними й вихідними афіними перетвореннями й логічними операціями. Шифр також використовує введення й вивід відбілювання клавіш. Шар дифузії використовує лінійне перетворення на основі матриці з номером галузей 5.

Аналіз безпеки

Камелія вважається сучасним надійним шифром. Навіть при використанні параметра меншого розміру ключа (128 біт) вважається неможливим зламати його за допомогою атаки грубої сили на ключі за допомогою сучасних технологій. Немає відомих успішних атак, що значно послабляють шифр. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		56

Пізніше, в 2008 році, група розробки релізу FreeBSD оголосила, що цей шифр також був включений в FreeBSD 6.4. Крім того, Йошисато Янагисава додав підтримку шифру Camellia у дисковий клас зберігання geli FreeBSD.

У вересні 2009 року GNU Privacy Guard додала підтримку Camellia у версії 1.4.10.

Veracrypt (відгалуження Truecrypt) включав Camellia як один з підтримуваних алгоритмів шифрування.

Крім того, різні популярні бібліотеки безпеки, такі як Crypto ++, Gnutls, mbed TLS і Openssl також включають підтримку Camellia.

26 березня 2013 р. було оголошено, що Camellia була знову обрана для включення в новий список рекомендованих шифрів для електронного уряду Японії як єдиний 128-бітний алгоритм блокового шифрування, розроблений у Японії. Це збігається з тим, що список CRYPTREC обновляється вперше за 10 років. Вибір був заснований на високій репутації Camellia у плані простоти придбання, а також характеристик безпеки й продуктивності, порівнянних з такими з Advanced Encryption Standard (AES). Камелія залишається незмінною у своєму повному втіленні. Нemoжлива диференціальна атака на Camellia з 12 раундами без шарів FL / FL дійсно існує.

Продуктивність

S-блоки, використовувані Camellia, мають структуру, аналогічну S-блоку AES. У результаті можна прискорити реалізацію програмного забезпечення Camellia за допомогою наборів команд ЦП, розроблених для AES, таких як x86 AES-NI.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		58

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення призначене для системи забезпечення цілісності даних у хмарних сервісах.

Програмно-апаратні вимоги:

– Загальний обсяг ОЗП: 512 Мбайт.
– Вільний простір на жорсткому диску серверу хмарного сервісу: 15 Мбайт.

– Операційна система Microsoft Windows 10/11.

– .Net Framework 2.0 (у Windows Vista він вже є).

Головне вікно програми зображене на рисунку 5.1.

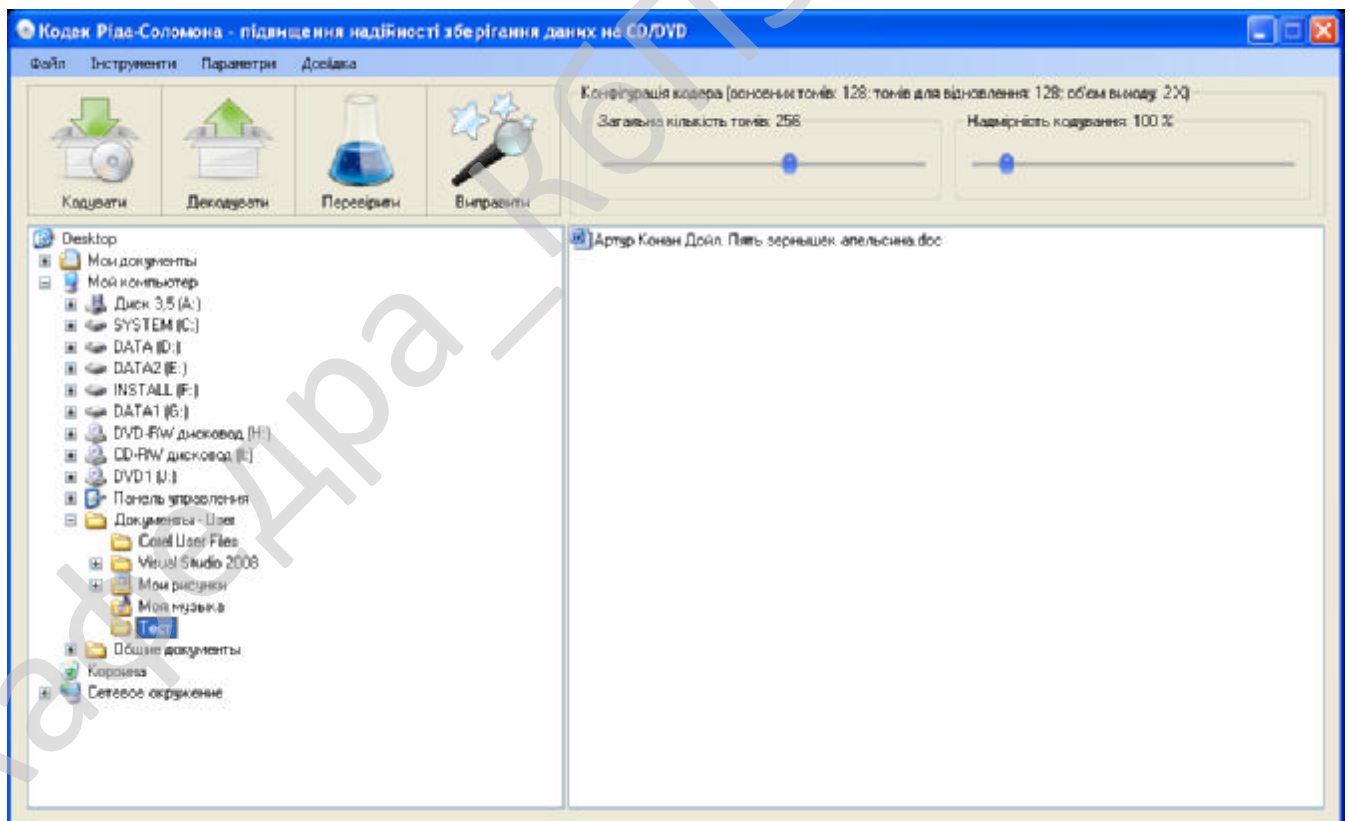


Рисунок 5.1 – Основне вікно програми

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		59

При надлишковому кодуванні файлу він розбивається на безліч фрагментів (томів). Потім на підставі основних томів обчислюються надлишкові, і для кожного файлу із сукупності розраховуються контрольні суми. При відновленні інформації декодер оцінює ушкодження наявного набору даних, і, якщо можливо, відновлює загублені основні томи з наступним об'єднанням у вихідний файл (який і являє собою відновлені дані).

Захист даних від ушкоджень

Насамперед, необхідно виділити директорію для обробки даних. Скопіюйте в неї всі файли, які Ви хочете закодувати у відмовостійкому форматі (при цьому слід пам'ятати, що **максимальна довжина імені файлу для кодування, становить 50 символів** (у підсумку вийде ім'я довжиною 64 символи, з додаванням префіксу, що обчислюється програмою). Потім виберіть зазначену директорію в оболонці програми, (робоча директорія вибирається в лівому вікні, при цьому оброблятися будуть ті файли, які видно в правому вікні) і натисніть на кнопку “Кодувати”. У вікні, що з'явилося (рисунок 5.2), Ви зможете спостерігати за процесом обробки даних.

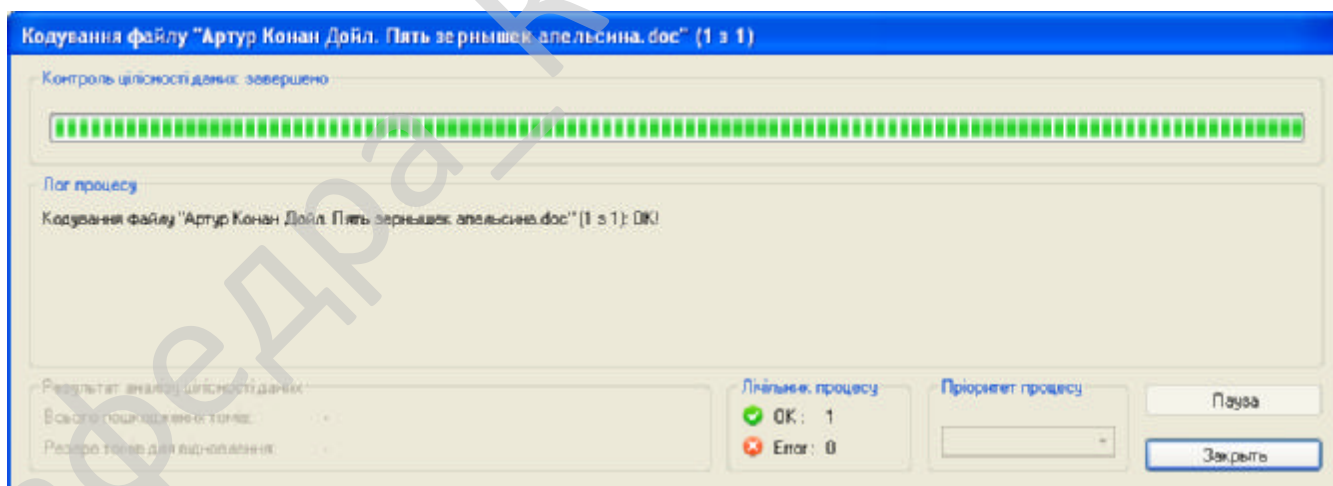


Рисунок 5.2 – Кодування файлу алгоритмом Ріда–Соломона

По-умовчанням загальна кількість томів – 256, але якщо Ви хочете підвищити захищеність даних, збільшіть загальну кількість томів (для потужних

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		60

машин рекомендується значення 1024). Надмірність же не рекомендується встановлювати нижче рівня в 50%. Пам'ятайте, що чим більшу кількість томів Ви створюєте, тим вище буде коригувальна здатність системи. Якщо ж Ви захочете записати кожний том у хмарних сервісах хмарний сервіс, доречними значеннями будуть 2,3,4 або 6. **При одночасному кодуванні й наступному записі на хмарний сервіс декількох файлів зростає максимальна коректуєма довжина безперервної послідовності ушкоджень.**

Отримана у результаті кодування множина томів показана на рисунку 5.3.

Після завершення обробки запишіть отриману множину файлів на хмарний сервіс (кожний такий файл – це том, що відноситься до свого архіву). **Для виділення множини файлів у поточній директорії можна використовувати комбінацію клавіш “Ctrl + A”.** Кожний том для відновлення заміняє собою один основний том у процесі виправлення помилок. Це означає, що якщо при кодуванні було створено 24 основні томи й 232 томи для відновлення, то для відновлення вихідної інформації буде досить 24 будь-яких файлів зі збереженого набору, у якому 256 файлів.

Декодування даних

Насамперед, необхідно виділити директорію для обробки даних. Скопіюйте в неї всі файли, які читаються приводом, виберіть зазначену директорію в оболонці програми (робоча директорія вибирається в лівому вікні, при цьому оброблятися будуть ті файли, які видно в правому вікні) і натисніть на кнопку “Декодувати”. У вікні, що з'явилося (рисунок 5.4), Ви зможете спостерігати за процесом обробки даних. Після закінчення обробки **всі відновлені файли будуть перебувати в робочій директорії, а всі вихідні томи будуть автоматично видалені.** Якщо в процесі обробки деякому файлу відповідає мітка “OK”, його дані можуть бути відновлені на 100%. Якщо файлу привласнюється мітка “Error” – файл не підлягає відновленню.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		61

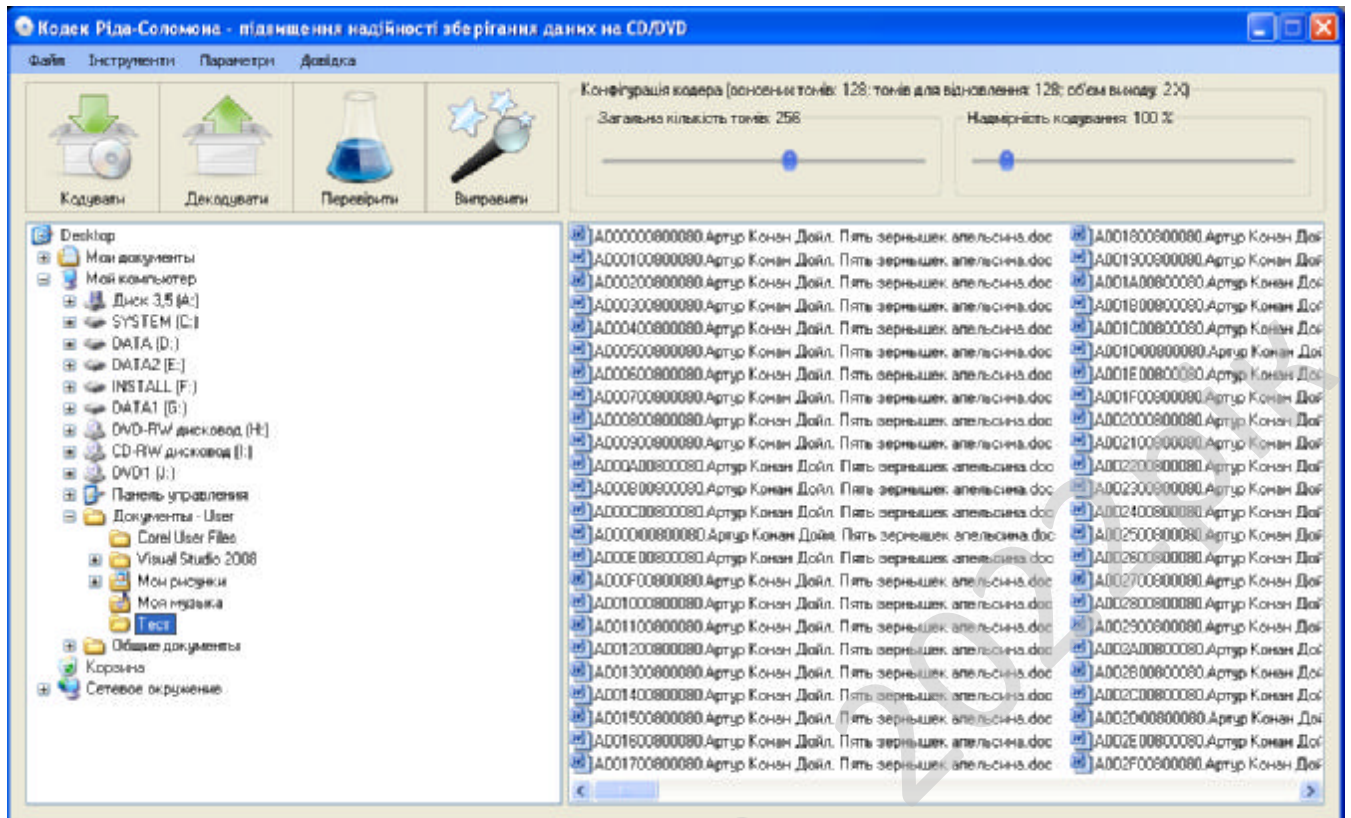


Рисунок 5.3 – Створені у результаті кодування томи для відновлення

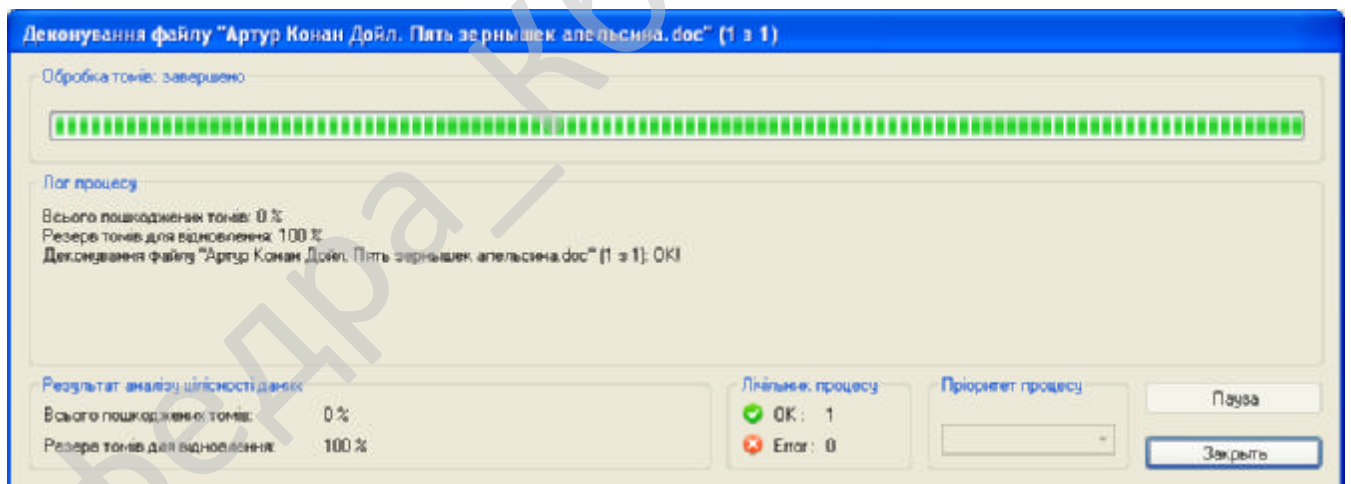


Рисунок 5.4 – Декодування файлу алгоритмом Ріда–Соломона

Перевірка даних на цілісність

Виберіть зазначену директорію в оболонці програми (робоча директорія вибирається в лівому вікні, при цьому оброблятися будуть ті файли, які видно в

правому вікні) і натисніть на кнопку “Перевірити”. У вікні, що з'явилося (рисунок 5.5), Ви зможете спостерігати за процесом аналізу даних. Якщо в процесі обробки деякому файлу відповідає мітка “OK”, його дані можуть бути відновлені на 100%. Якщо файлу привласнюється мітка “Error” – файл не підлягає відновленню.

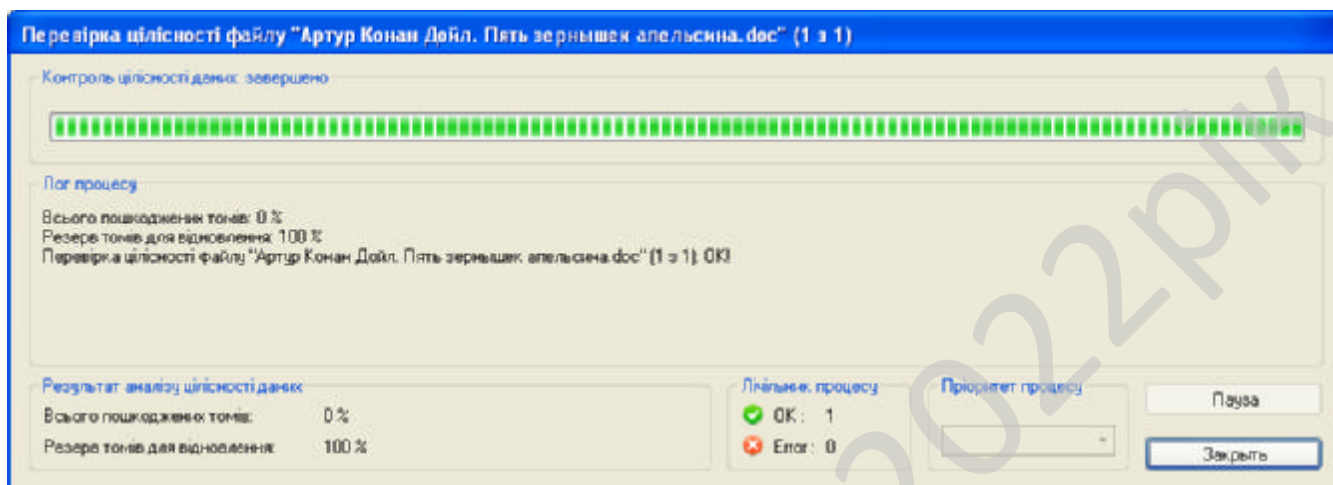


Рисунок 5.5 – Перевірка цілісності файлу

Відновлення пошкоджених даних

Насамперед, необхідно виділити директорію для обробки даних. Скопіюйте в неї всі файли, які читаються приводом, виберіть зазначену директорію в оболонці програми і натисніть на кнопку “Відновити”. У вікні, що з'явилося (рисунок 5.6), Ви зможете спостерігати за процесом обробки даних. Після закінчення обробки в робочій директорії буде перебувати вихідна множина томів, що відповідає стану після кодування вихідних даних (тобто всі знищені або ушкоджені томи будуть відновлені). Якщо в процесі обробки деякому файлу відповідає мітка “OK”, його дані можуть бути відновлені на 100%. Якщо файлу привласнюється мітка “Error” – файл не підлягає відновленню.

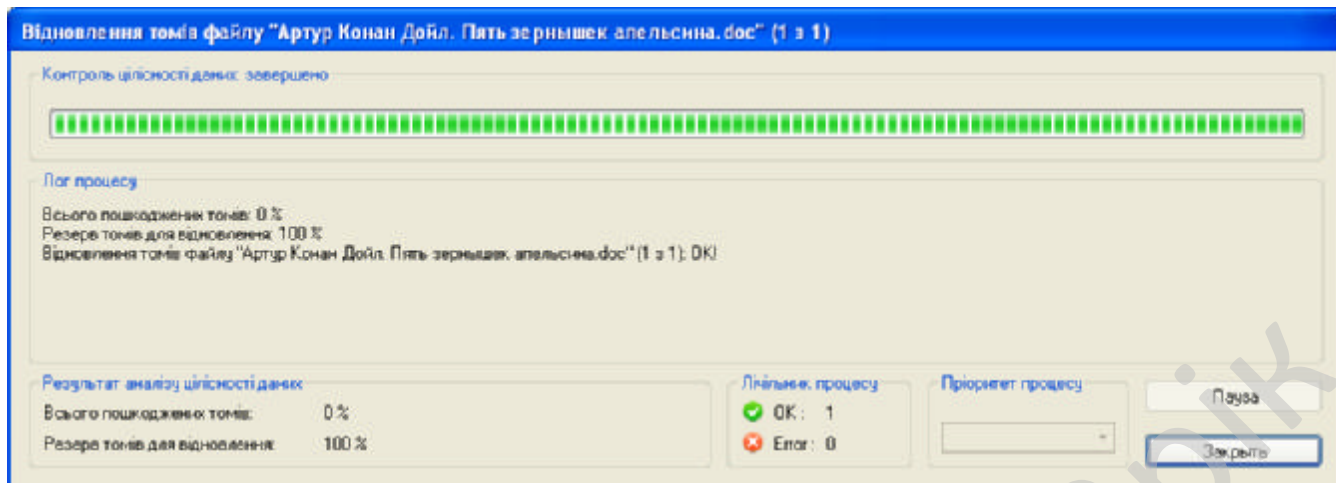


Рисунок 5.6 – Відновлення пошкоджених томів

Розроблена програма здатна захищати дані від несанкціонованого доступу за допомогою алгоритму AES-256 через шифруючий фільтр, що може бути активований через пункт меню “Настроювання”→Шифруючий фільтр. Після вибору даного пункту меню з'являється вікно, зображене на рисунку 5.7, у якому слід вказати пароль та розмір CBC-блоку.

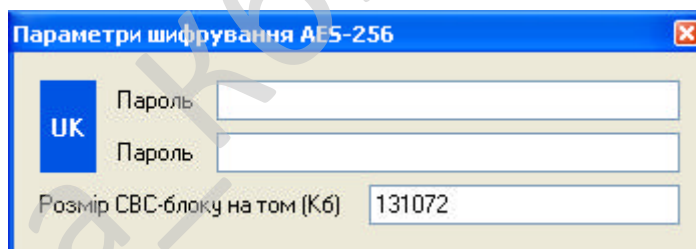


Рисунок 5.7 – Встановлення паролю на закодовані дані

Тестування швидкодії кодека Ріда-Соломона

Програма має вбудований тест швидкодії кодека Ріда-Соломона. Для запуску тесту виберіть підпункт “Тест швидкодії” з меню “Інструменти”. Після цього з'явиться вікно зображене на рисунку 5.8, що містить результати тесту.

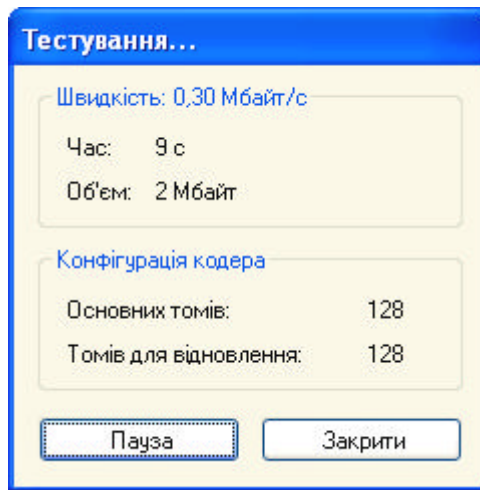


Рисунок 5.8 – Тестування швидкості роботи алгоритму Ріда–Соломона

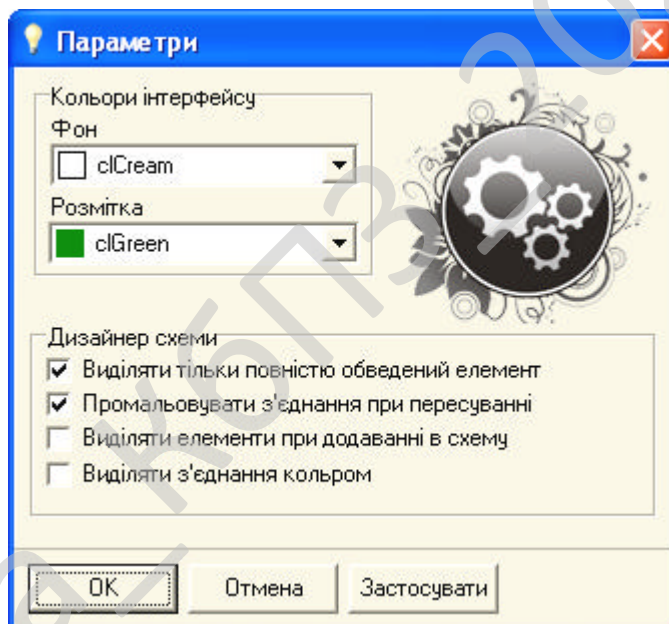


Рисунок 5.9 – Задача параметрів інтерфейсу

Коротку довідку про розроблену програму та її автора можна переглянути вибравши підпункт “Про програму...” з меню “Довідка”, після чього з'явиться вікно зображене на рисунку 5.10.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Центральноукраїнський національний технічний
університет
Кафедра кібербезпеки та програмного забезпечення
ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему: "Дослідження та програмна реалізація системи
забезпечення цілісності даних у хмарних сервісах"
ОПП «Комп'ютерні науки»
спеціальності 122 «Комп'ютерні науки»
Виконав: Мицак І.М.
Науковий керівник: Смірнов С.А.
Кропивницький - 2022

ок

Рисунок 5.10 – Довідка про програму

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		66

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи забезпечення цілісності даних у хмарних сервісах.

Метою розробки є дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах.

Об'єктом дослідження є процес забезпечення цілісності даних у хмарних сервісах.

Предметом дослідження є методи забезпечення цілісності даних у хмарних сервісах.

Методи дослідження базуються на методах забезпечення цілісності даних, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

- Удосконалено метод забезпечення цілісності даних у хмарних сервісах.
- Розроблено вітчизняний продукт забезпечення цілісності даних у хмарних сервісах, який має більш широкі можливості, на відміну від існуючих аналогів.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		67

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи забезпечення цілісності даних у хмарних сервісах.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

- а) невеликий розмір;
- б) невеликі системні потреби;
- в) незалежність від встановлених на комп'ютері баз даних;
- г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	120
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	120000
33. Норматив додаткової зарплати, % :	Нд	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \text{ Size}^B, \quad (7.1)$$

де: A – коефіцієнт Боема, $A = 2,45$;

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		70

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1,01 + 0,001 \sum W_i, \quad (7.2)$$

де: W_i – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1,01 + 0,001(2,43 + 3,64 + 3,38 + 3,95 + 2,73) = 1,027.$$

$$T_{ном} = 2,45 \cdot 2,7^{1,026} = 6,78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном} PV_j, \quad (7.3)$$

де: PV_j – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6,78 \cdot (0,88 \cdot 0,93 \cdot 0,88 \cdot 0,91 \cdot 0,95 \cdot 1 \cdot 1 \cdot 0,87 \cdot 1,22 \cdot 1,16 \cdot 1,1 \cdot 1,1 \cdot 1,12 \cdot 1,1 \cdot 1,1 \cdot 1,1) = 9,37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0,3 C T_{уточн}^{0,33+0,2(B-1,01)} S, \quad (7.4)$$

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25...350)%.

$$T_{РП} = 0,3 \cdot 2,66 \cdot 9,37^{0,33+0,2(1,026-1,01)} \cdot 120 = 202 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		71

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	202	Ф 7.1-7.4
Впровадження	13	Д13
Всього	243	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{nz} N}{F_{pq} - H_{ев}}, \quad (7.5)$$

де: F_{pq} – плановий фонд робочого часу одного спеціаліста, днів;

T_{nz} – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{243 \cdot 1}{60 - 5} = 4,4 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість обладнання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	13	1170	19,5
Монітор	60	13	780	13
Клавіатура	30	13	390	6,5
Маніпулятор «мишка»	30	13	390	6,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	2	40	0,67
Концентратор-маршрутизатор	30	3	90	1,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	250	625	10,42
Копіювальний апарат	140	1	140	2,33
Усього за рік:			3 _ц	65,42

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків не повинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$\Phi_{op}^c = \frac{3_{ц} \cdot n_{mic}}{1,2}, \quad (7.6)$$

$$\Phi_{op}^c = \frac{65,4 \cdot 3}{1,2} = 163,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{\Phi_{др}^c}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 163,5 / (60 \cdot 8) = 0,3 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (OC FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (OC Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпроводного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	6970	20910
Продакт-менеджер	0,25	6700	5025
Інженер-програміст	4,4	7000	92400
Інженер-електронщик	0,3	6700	6030
Інженер-системотехнік	0,25	6700	5025
Адміністратор мережі	0,5	6700	10050
Системний програміст	0,25	6700	5025
Дизайнер WEB	0,25	6700	5025
Інженер-верстальник	0,25	6700	5025
Бухгалтер-економіст	0,5	6700	10050
Всього за період розробки	$R_{cn} = 7,95$	-	$\Phi_{роб} = 164565$

Розрахуємо середньоденну зарплату одного виконавця:

$$z_{cd} = \frac{\Phi_{роб}}{R_{cn} F_{pq}}, \quad (7.8)$$

де: $\Phi_{роб}$ – загальна сума зарплати за плановий період, грн.

$$z_{cd} = \frac{164565}{8,25 \cdot 60} = 345 \text{ грн.}$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		76

$$B_{y\partial} = R_{cn}^1 S_y C_{nl}, \quad (7.9)$$

де: R_{cn}^1 – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

S_y – питома площа на одне робоче місце, m^2 ;

C_{nl} – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (<https://dom.ria.com>) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500...1600 у.о./ m^2 . Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./ m^2 . На кожне робоче місце у середньому потрібно $8 m^2$. З урахуванням цього:

$$B_{y\partial} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$I_{nb} = R_{cn}^1 \cdot C_m, \quad (7.10)$$

де: C_m – ціна меблів для одного робочого місця, грн.

$$I_{nb} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет магазину Компбест за 06.11.22 – джерело <https://compbest.com.ua>.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		10947

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системний блок		7347
Процесор	Intel Core i7 860 8*3.5GHz/8MB/2.5GTs 87s1156, box	1750
Системна плата	MB MSI H55M-E33 s1156 mATX (H55ME33)	1200
Відеокарта	VC VTX Radeon HD6570 1GB GDDR3 128bit, 650 MHz/1334 MHz, PCI-E 2.1, DV HDMI, VGA (VX6570 1GBK3-H)	750
Жорсткий диск	HDD: 320 Gb 7200 Serial ATA WD 16MB	1200
Оперативна пам'ять	DIMM 2048Mb DDR3 PC3-12800 Patriot 1600Mhz, CL9, (9-9-9-28), 1.5V, Retail (PSD32G16002H) 2 модуля	900
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slim Black Bulk (DVR-TD10RS)	416
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA489, PSU 350W(FSP Brand: ATX-350PNR 12cm), black, (front bezel – black+light silver body material – 0.6mm), 80mm fan (rear) 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-less chassis design,Thermally Advantaged Chassis	911

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	220
Інше	Клавіатура, мишка	Подарунок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	0	20	0,0
6. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
7. Нематеріальні активи	120000	10	12000
Разом	$K_p = 1760367$		$A_p = 190286$

Згідно прийнятих норм на підприємстві $n_{\text{вип}}$ приймаємо 0,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_n=210$ грн., визначаємо вартість паперу за період розробки:

$$З_{M1} = Ц_n \cdot N_m. \quad (7.16)$$

$$З_{M1} = 210 \cdot 0,5 = 105 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуваних пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 120):

$$З_{M2} = \sum Ц_d, \quad (7.17)$$

де: $Ц_d$ – вартість дисків CD/DVD: – 12 грн./шт.

$$З_{M2} = 120 \cdot 12 = 1440 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{M3} = \sum Ц_z, \quad (7.18)$$

де: $Ц_z$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{M3} = 574 + 558 + 570 = 1702 \text{ грн.}$$

$$З_M = (105 + 1440 + 1702) / 120 = 27 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($H_n = 15\%$) від основної зарплати виконавців:

$$O_n = З_o \cdot H_n \cdot 0,01, \quad (7.19)$$

де: H_n – норматив витрат на освоєння нових мов програмування, %.

$$O_n = 698 \cdot 15 \cdot 0,01 = 105 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_e = 120$ прим.):

$$A_m = \frac{A_p \cdot N_{\text{міс}}}{N_e \cdot 12}, \quad (7.20)$$

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		82

де: A_p – загальна річна сума амортизаційних відрахувань, грн.

$$A_m = 190286 \cdot 3 / (120 \cdot 12) = 396 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_n = Z_o + Z_d + C_{oc} + \Gamma_{ocn} + Z_m + O_n + A_m. \quad (7.21)$$

$$C_n = 698 + 70 + 168 + 105 + 27 + 105 + 396 = 1570 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1. Основна зарплата виконавців	Z_o	698
2. Додаткова зарплата виконавців	Z_d	70
3. Відрахування на соціальні потреби	C_{oc}	169
4. Загальногосподарські витрати	Γ_{ocn}	105
5. Витрати на матеріали	Z_m	27
6. Освоєння нових операційних систем, мов програмування	O_n	105
7. Амортизація основних фондів	A_m	396
8. Повна собівартість програмного забезпечення	C_n	1570
9. Плановий прибуток	P_p	785
10. Ціна підприємства $C_n = C_n + P_p$	C_n	2355
11. Податок на додану вартість $ПДВ = 0.01 \cdot H_{ос} \cdot C_n$	$ПДВ$	471
12. Відпускна ціна програмної продукції $C = C_n + ПДВ$	C	2826

Визначимо плановий прибуток за рівнем рентабельності (P_n) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$P_p = 0,01 \cdot P_n \cdot C_n, \quad (7.22)$$

де: P_n – рівень рентабельності, %.

$$P_p = 0,01 \cdot 50 \cdot 1570 = 785 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконаладжувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	2826
Всього капітальних витрат	–	2826

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	Z_p	15190	3500
2. Витрати на електроенергію	$Z_{ел}$	227	157
3. Витрати на амортизацію	$Z_{ам}$	0	707
Всього витрат за рік	I	15417	4364

Витрати на профілактичні роботи:

$$Z_p = T_p \cdot Z_2 \cdot (1 + 0,01 \cdot H_q) \cdot (1 + 0,01 \cdot H_c), \quad (7.23)$$

де: T_p – кількість годин обслуговування кожного комп'ютера за рік, год.;

Z_2 – заробітна плата обслуговуючого персоналу, грн/год.

Після купівлі нового програмного забезпечення витрати на обслуговування системи з 15190 грн до 3500 грн на рік.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	2826	–	706,5
Всього відрахувань	-	–	2826	–	706,5

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($P_{ел}$) в кіловатах, часу експлуатації технічних засобів (T_p) в годинах та ціни однієї кіловат-години ($C_{ел}$):

$$Z_{ел} = P_{ел} \cdot T_p \cdot C_{ел} \quad (7.24)$$

$$Z_{ел \text{ баз}} = 0,545 \cdot 130 \cdot 3,2 = 227 \text{ грн.}$$

$$Z_{ел \text{ нов}} = 0,545 \cdot 90 \cdot 3,2 = 157 \text{ грн.}$$

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_e = (C_n - C_n) \cdot N_e - \sum_{i=1}^m E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: K_p – балансова вартість основних фондів розробника, грн.; E_p – розрахунковий коефіцієнт капіталовкладень.

$$E_e = (2355 - 1570) \cdot 120 - (0,05 \cdot 1408000 + 0,4 \cdot 199177 + 0,25 \cdot 33190 + 0,1 \cdot 120000) \cdot 3/12 = 51608 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_e = \frac{K_p}{(C_n - C_n) \cdot N_e}, \quad (7.26)$$

де: K_p – балансова вартість основних фондів розробника.

$$T_e = \frac{1760367}{(2355 - 1570) \cdot 120 \cdot 12 / 3} = 4,6 \text{ роки}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{cn} = (I_o - I_n) - E_n(K_n - K_o), \quad (7.27)$$

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		86

де: I_{δ} , I_n – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

K_{δ} , K_n – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{en} = (15417 - 4364) - 0,25 \cdot 2826 = 10347 \text{ грн.}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	120
2. Повна собівартість розробленої програми	Грн.	1570
3. Ціна розробленої програми	Грн.	2355
4. Плановий прибуток від реалізації розробленої програми	Грн.	785
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1760367
7. Загальний прибуток від реалізації програмної продукції	Грн.	94200
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	51608
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	4,6
10. Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	2826
11. Величина економічного ефекту у користувача програмної продукції	Грн.	10347
12. Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,26

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{cn} = \frac{K_n - K_b}{I_b - I_n}, \quad (7.28)$$

$$T_{cn} = \frac{2826}{15417 - 4364} = 0,26 \text{ року.}$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		88

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона праці – це: система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини в процесі трудової діяльності;

Охорона праці є складовою частиною безпеки життєдіяльності [3,4].

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		89

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

- параметри повітряного середовища в приміщенні;
- вентиляція приміщення;
- освітлення приміщення;
- параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

Загальний нагляд за додержанням норм охорони праці покладено на прокуратуру, спеціальний покладено на професійні спілки. За безпекою контроль праці здійснюють державні й відомчі спеціалізовані інспекції.

У Законодавстві про працю міститься вимоги і норми з виробничої санітарії, техніки безпеки та норми, що регулюють робочий час, час відпочинку, звільнення та переведення на іншу роботу, а також норми праці щодо жінок, молоді, гігієнічні норми і правила, тощо.

8.2 Аналіз умов праці

Фірма дотримується всіх правил з охорони праці і слідкує за їх дотриманням.

При виконанні робіт на комп'ютерах працівникам необхідно дотримуватись вимог загальної інструкції з охорони праці.

До роботи на комп'ютерах допускаються особи, які пройшли: медичний огляд, навчання по професії, вступний інструктаж з охорони праці та первинний

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		90

інструктаж з охорони праці на робочому місці. В подальшому вони проходять повторні інструктажі з охорони праці на робочому місці один раз на півріччя, періодичні медичні огляди один раз на два роки.

Основним обладнанням робочого місця є монітор, системний блок, миша та клавіатура.

Робочі місця розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін на відстані 1 м та між собою на відстані не менше 1,5 м. Відносно вікон робоче місце доцільно розташовувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

Монітор розташований на робочому місці так, щоб поверхня екрана знаходилася в центрі поля зору на відстані 400-700 мм від очей користувача. Елементи робочого місця розміщуються так, щоб витримувалася однакова відстань очей від екрана, клавіатури, тексту.

Джерела освітлення розташовані з обох боків екрану паралельно напрямку погляду. Для уникнення світлових відблисків екрану, клавіатури в напрямку очей користувача, від світильників загального освітлення або сонячних променів, також використовують антиблікові сітки, спеціальні фільтри для екранів, захисні козирки, на вікнах – жалюзі.

Використовуються скляні поляризаційні фільтри вони забезпечують найкращу якість зображення. Вони усувають практично всі відблиски, роблять зображення чітким і контрастним.

Зручна робоча поза при роботі з комп'ютером забезпечується регулюванням висоти робочого столу, крісла та підставки для ніг. Працівники мають пам'ятки, що раціональною робочою позою вважається положення, при якому ступні працівника розташовані горизонтально на підлозі або підставці для ніг, стегна зорієнтовані у горизонтальній площині, верхні частини рук – вертикальні. Кут ліктьового суглоба коливається в межах 70 – 90°, зап'ястя зігнуті під кутом не більше ніж 20°, нахил голови 15 – 20°.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		91

– необхідно стійко розташовувати клавіатуру на робочому столі, не опускати її хитання. Під час роботи на клавіатурі сидіти прямо, не напружуватися;

– для забезпечення несприятливого впливу на користувача пристроїв типу "миша" належить забезпечувати вільну велику поверхню столу для переміщення "миші" і зручного упору ліктя суглоба;

– не дозволяються сторонні розмови, подразнюючі шуми;

– періодично при вимкненому комп'ютері прибирати ледь змоченою мильним розчином бавовняною ганчіркою порох з поверхонь апаратури. Екран ВДТ та захисний екран протирають ганчіркою, змоченою у спирті. Не дозволяється використовувати рідинні або аерозольні засоби чищення поверхонь комп'ютера.

Працівники не повинні порушувати правил з охорони праці та їм забороняється:

– самостійно ремонтувати апаратуру. Ремонт апаратури здійснюється спеціалістами з технічного обслуговування комп'ютера, 1 раз на півроку повинні відкривати процесор і вилучати пиломасою пил і бруд, що накопичилися;

– класти будь-яку предмети на апаратуру комп'ютера;

– закривати будь-чим вентиляційні отвори апаратури, що може призвести до її перегрівання і виходу з ладу.

Для зняття статичної електрики рекомендується час від часу доторкатися до металевих поверхонь.

Розташувати принтер необхідно поруч з системним блоком таким чином, щоб з'єднувальний шнур не був натягнутий. Забороняється ставити принтери на системний блок.

Для досягнення найбільш чистих, з високою роздільністю зображень і щоб не зіпсувати апарат, має використовуватися папір, вказаний в інструкції до принтера. При змінанні паперу потрібно відкрити кришку і обережно витягнути лоток з папером.

Працівника потрібно дотримуватися вимоги безпеки після закінчення роботи:

- закінчити та записати у пам'ять комп'ютера файл, що знаходиться в роботі;
- вимкнути принтер та інші периферійні пристрої. Штепсельні вилки витягнути з розеток. Накрити клавіатуру кришкою запобігання попаданню в неї пилу;
- прибрати робоче місце;
- ретельно вимити руки теплою водою з милом;
- вимкнути кондиціонер, освітлення і загальне електроживлення;
- пройти в спеціально обладнаному приміщенні сеанс психофізіологічного розвантаження і зняття втоми з виконанням спеціальних вправ аутогенного тренування.

8.4 Пожежна безпека

Пожежі в приміщеннях з оргтехнікою становлять особливу небезпеку, бо поєднані з великими матеріальними збитками. Пожежа може виникнути при взаємодії горючих речовин і джерел запалювання. Горючими речовинами є будівельні та опоряджувальні матеріали, пластмасові корпуси техніки, шнури тощо. Джерелами запалювання можуть бути електронні схеми комп'ютерів, принтерів, пристроїв електроживлення, де внаслідок різних порушень виникає перегрівання елементів, утворюються електричні іскри та дуги, здатні спричинити займання горючих матеріалів.

З метою виявлення початкової стадії займання необхідно використовувати пристрої систем автоматичного пожежогасіння там, де цього вимагають правила пожежної безпеки.

При обслуговуванні, ремонтних та профілактичних роботах використовуються різні легкозаймісті рідини, прокладаються тимчасові

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		95

відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін} = 50\%$ і $\rho_{стелі} = 50\%$.

Обчислимо індекс приміщення за формулою:

$$i = S / (h(A+B)),$$

де:

S – площа приміщення, $S = 6,76 \text{ м}^2$;

h – розрахункова висота підвісу, $h = 3 \text{ м}$ (співпадає з висотою стелі, т.я. лампи освітлення закріплюються на стелі);

A – ширина приміщення, $A = 2,6 \text{ м}$;

B – довжина приміщення, $B = 2,6 \text{ м}$.

Підставимо всі значення у формулу та визначимо індекса приміщення:

$$i = 0,43.$$

Знаючи індекс приміщення, за знаходимо $n = 0,23$ (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам). Підставимо всі значення у формулу, визначимо світловий потік: $F = 14548 \text{ Лм}$.

Будемо використовувати світлодіодні панелі Lebron L-LPU-Prismatic, світловий потік яких $F_n = 3000 \text{ Лм}$.

Число ламп визначається по формулі:

$$N = F / F_n$$

де:

F – світловий потік,

F_n – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначимо індекса приміщення:

$$N = 14548 / 3000 = 4,8 \text{ шт.}$$

Приймаємо необхідну кількість ламп 5 шт.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		98

8.6 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		99

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи забезпечення цілісності даних у хмарних сервісах.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів забезпечення цілісності даних у хмарних сервісах.

Рішення даного завдання полягало у вирішенні наступних задач:

- Був проведений огляд існуючих систем забезпечення цілісності даних у хмарних сервісах.
- Досліджена система забезпечення цілісності даних у хмарних сервісах.
- На основі отриманих результатів досліджень створена програмна реалізація системи забезпечення цілісності даних у хмарних сервісах.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання забезпечення цілісності даних у хмарних сервісах.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		100

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

Програма реалізована на мові високого рівня Visual C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати термін 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 Camellia.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 10347 грн. З урахуванням вартості розробки програми та обладнання, термін окуплення становить 0,26 роки.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		101

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мицак І.М. Дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

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

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

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

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

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

7. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th*

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		102

International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus)**.

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

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

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

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

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

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

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

15. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions».

Lecture Notes in Networks and Systems, vol 152. **Springer**, Cham. 2021, pp 66-84. **(Scopus)**.

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. **(Scopus)**.

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. **(Scopus)**.

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. **(Scopus)**.

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

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

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

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

23. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during

Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019. **(Scopus)**.

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. **(Scopus)**.

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

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

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

28. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: *ISCI'2019: Information Security in Critical Infrastructures. Collective monograph*. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

29. 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).

30. 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.

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

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

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

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. 2022. (**Фахове видання. Категорія «Б»**)

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

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		106

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

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

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

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

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

43. 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.

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		107

44. 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.

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

46. 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.

47. О.А. Смірнов, Т.В. Смірнова, О.М. Дреєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.

48. Смірнов О.А., Дреєва Г.М., 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.

49. Смірнов О.А., Смірнова Т.В., Солових Є.К., 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.

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

51. 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.

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

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

54. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: <https://goo.su/9AkQ>

55. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: <https://zakon.rada.gov.ua/rada/show/v0007282-98>

56. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-ХІІ. – Режим доступу до ресурсу: <https://zakon.rada.gov.ua/laws/show/2694-12>

57. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

58. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: <https://zakon.rada.gov.ua/laws/show/z0508>

59. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ IBM сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград:

					ВКРМ-122.22.0007.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		109

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 Економічні вимоги.....	5
8 Вимоги щодо охорони праці.....	5
9 Перелік документів, що розробляються.....	6
10 Етапи розробки.....	6
11 Порядок контролю та приймання.....	6

					ВКРМ-122.22.0007.00.00.ТЗ			
Вим.	Арк.	№ документа	Підпис	Дата				
Розробив	Мицак І.М.				Дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах	Літ.	Аркуш	Аркушів
Перевірів	Смірнов С.А.					М	1	6
Н. Контр.	Гермак В.С.				ЦНТУ КН-21М-1,4			
Затв.	Смірнов О.А.							

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи забезпечення цілісності даних у хмарних сервісах.

2 Підстава для розробки

Підставою для розробки служить завдання на випускню кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 18-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи забезпечення цілісності даних у хмарних сервісах.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

- вибір і обґрунтування методів реалізації проекту;
- розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

					ВКРМ-122.22.0007.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		2

- техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;
- аналіз умов праці;
- розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

- програмну реалізацію системи забезпечення цілісності даних у хмарних сервісах;
- 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.

					ВКРМ-122.22.0007.00.00.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 архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Visual C#.

					ВКРМ-122.22.0007.00.00.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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці.

					ВКРМ-122.22.0007.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		5

9 Перелік документів, що розробляються

- Наукова новизна – 1 аркуш.
- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Показники економічної ефективності – 1 аркуш.
- Пояснювальна записка – 110 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2022 р.

					ВКРМ-122.22.0007.00.00.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
другим (магістерським) рівнем вищої освіти
_____ Смірнов С.А.

*Дослідження та програмна реалізація
системи забезпечення цілісності даних у хмарних сервісах*

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 61

Літера: РП

Кропивницький – 2022 року

Кодер Ріда-Соломона

Файл RSRaidEncoder.cs - кодування алгоритмом Ріда-Соломона

```

using System;
using System.Threading;

namespace RecoveryDisk
{
    /// <summary>
    /// Клас RAID-подібного кодера Ріда-Соломона
    /// </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 Ріда-Соломона (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 кодека Ріда-Соломона (по типу
матриці)</param>
    /// <returns>Булевський прапор операції установки конфігурації</returns>
public bool SetConfig(int dataCount, int eccCount, int codecType)
{
    int maxVolCount;

    // Установлюємо константи, що відповідають обраному режиму
    if (codecType == (int)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 Ріда-Соломона

Файл RSRaidDecoder.cs - декодування алгоритмом Ріда-Соломона

```

using System;
using System.Threading;

namespace RecoveryDisk
{
    /// <summary>
    /// Клас RAID-подібного декодера Ріда-Соломона
    /// </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 Ріда-Соломона (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 кодека Ріда-Соломона (по типу
матриці)</param>
    /// <returns>Булевський прапор операції установки конфігурації</returns>
    public bool SetConfig(int dataCount, int eccCount, int[] volList, int
codecType)
    {
        int maxVolCount;

        // Установлюємо константи, що відповідають обраному режиму
        if (codecType == (int)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);

```

```

стадії // Кількість ітерацій, що відслідковуються прогресом, на першій

// 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
            int j;

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

            data[i] = mulSum;

```

стовпець
рядка

```

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

    return true;
}

#endregion Public Operations

#region Private Operations

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

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

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

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

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

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

```

```

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

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

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

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

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

    return false;
}

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

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

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

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

        int idx = i_n + j;

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

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

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

```

```

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

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

    // Якщо є передплата на делегата відновлення прогресу -...
    if (
        ((k % progressMod1) == 0)
        &&
        (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;
}

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

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

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

/// <summary>
/// Заповнення матриці "FLog" (матриці декодера) даними

```

```

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

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

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

        return;
    }

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

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

    //...і вектор 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
        "A"
        if (!MakeAlternativeMatrix())
        {
            // Указуємо, що кодер зконфігуровано некоректно
            this.configIsOK = false;

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

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

            return;
        }
    }

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

// Для кожного загубленого основного тому шукаємо том для
відновлення

```

```

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

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

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

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

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

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

        // Указуємо, що даний рядок матриці "FLog" не тривіальна
        this.FLogRowIsTrivial[i] = false;
    } else
    {
        // Формуємо в матриці "FLog" нульовий рядок з одиницею на
головній діагоналі
        // (відповідає наявному основному той)
        DRowIdx = i;

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

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

        // Формування рядка в матриці кодування
        // ("тривіальні" рядки вже втримуються в матриці "D", вони
вийшли
        // "автоматично" на попередньому етапі обробки
        MakeDispersal()
        for (int j = 0; j < this.n; j++)
        {

```

```

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

        this.FLog[i_n + i] = 1;
    } else
    {
        int bs = (DRowIdx - this.n) * this.n;

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

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

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

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

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

    return;
}

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

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

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

    return;
}

// Обчислюємо логарифми елементів інвертованої матриці
LogFCalc();

// Якщо є передплата на делегата завершення...
if (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
}
}
```

Кафедра _ КБПЗ _ 2022 рік

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

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

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

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

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

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

    #endregion Data

    #region Construction & Destruction

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

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

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

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

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

```

```

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

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

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

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

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

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

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

#endregion Construction & Destruction

#region Public Operations

/// <summary>
/// Метод запуску потоку обробки обчислення й записи CRC64 у кінець
файлів
/// </summary>
/// <param name="path">Шлях до файлів для обробки</param>
/// <param name="fileName">Ім'я файлу для обробки</param>
/// <param name="dataCount">Конфігурація кількості основних
томів</param>
/// <param name="eccCount">Конфігурація кількості томів для
відновлення</param>
/// <param name="codecType">Тип кодека Ріда-Соломона (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 Ріда-Соломона (по типу використовуваної
    матриці кодування)
    this.codecType = codecType;

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

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

        // Повертаємо результат обробки
        return this.processedOK;
    }

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

    //...потім даємо йому ім'я...
    this.thrFileAnalyzer.Name = "FileAnalyzer.WriteCRC64()";

```

```

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

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

// Повідомляємо, що все нормально
return true;
}

/// <summary>
/// Метод запуску потоку обробки перевірки CRC64, записаного в кінець
/// кожного з файлів набору, з генеруванням списку наявних томів
"volList",
/// який буде використаний декодером для відновлення даних
/// </summary>
/// <param name="path">Шлях до файлів для обробки</param>
/// <param name="fileName">Ім'я файлу для обробки</param>
/// <param name="dataCount">Конфігурація кількості основних
томів</param>
/// <param name="eccCount">Конфігурація кількості томів для
відновлення</param>
/// <param name="codecType">Тип кодека Ріда-Соломона (по типу
матриці)</param>
/// <param name="fastExtraction">Використовується швидке добування з
томів (без перевірки CRC-64)?</param>
/// <param name="runAsSeparateThread">Запустити 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 Ріда-Соломона (по типу використовуваної
    матриці кодування)
    this.codecType = codecType;

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

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

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

        // Повертаємо результат обробки
        return this.processedOK;
    }

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

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

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

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

```

```

        // Повідомляємо, що все нормально
        return true;
    }

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

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

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

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

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

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

    #endregion Public Operations

    #region Private Operations

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

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

        // Піддаємо обробці всі томи
        for (int volNum = 0; volNum < (this.dataCount + this.eccCount);
        volNum++)
        {
            // Зчитуємо первісне ім'я файлу
            String fileName = this.fileName;

```

```

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

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

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

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

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

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

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

            continue;
        }

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

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

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

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

            return;
        }

        //...якщо одержали сигнал про завершення обробки
        // вкладеним алгоритмом...

```

```

        if (eventIdx == 2)
        {
            //...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",
        // щоб прогрес виводився на кожній ітерації (файл дуже маленький)
        if (progressMod1 == 0)
        {
            progressMod1 = 1;
        }

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

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

        // Індекс у масиві томів
        int volListIdx = 0;

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

        // Лічильник кількості ушкоджених основних томів
        int dataVolMissCount = 0;

```

```

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

// Ім'я файлу для обробки
String fileName;

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

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

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

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

    // Якщо вихідний файл існує...
    if (File.Exists(fileName))
    {
        // Якщо не використовується швидке добування - перевіряємо
на цілісність
        // CRC-64, інакше думаємо, що все 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)!) break;

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
                    // на паузу -
                    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
                        // 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
                    // переходимо на нову ітерацію, тому що
                    // 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;
    }
}

// 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 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 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 відмовостійкого набору 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 Ріда-Соломона - 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;
    internal 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.AutomaticDelay = 2000;
this.toolTip.AutoPopDelay = 20000;
this.toolTip.InitialDelay = 2000;
this.toolTip.ReshowDelay = 1000;
//
// stopButtonXP
//
this.stopButtonXP.BackColor =
System.Drawing.Color.FromArgb(((int)((byte)(0))), ((int)((byte)(236))),
((int)((byte)(233))), ((int)((byte)(216))));
this.stopButtonXP.DefaultScheme = true;
this.stopButtonXP.DialogResult =
System.Windows.Forms.DialogResult.None;
this.stopButtonXP.Hint = "";
this.stopButtonXP.Location = new System.Drawing.Point(762, 257);
this.stopButtonXP.Name = "stopButtonXP";
this.stopButtonXP.Scheme = PinkieControls.ButtonXP.Schemes.Blue;
this.stopButtonXP.Size = new System.Drawing.Size(121, 23);
this.stopButtonXP.TabIndex = 2;
this.stopButtonXP.Text = "Перервати обробку";
this.toolTip.SetToolTip(this.stopButtonXP, "Припинення обробки
файлів із закриттям даного вікна");
this.stopButtonXP.Click += new
System.EventHandler(this.stopButtonXP_Click);
//
// pauseButtonXP
//
this.pauseButtonXP.BackColor =
System.Drawing.Color.FromArgb(((int)((byte)(0))), ((int)((byte)(236))),
((int)((byte)(233))), ((int)((byte)(216))));
this.pauseButtonXP.DefaultScheme = true;
this.pauseButtonXP.DialogResult =
System.Windows.Forms.DialogResult.None;
this.pauseButtonXP.Hint = "";
this.pauseButtonXP.Location = new System.Drawing.Point(762, 220);
this.pauseButtonXP.Name = "pauseButtonXP";
this.pauseButtonXP.Scheme = PinkieControls.ButtonXP.Schemes.Blue;
this.pauseButtonXP.Size = new System.Drawing.Size(121, 23);
this.pauseButtonXP.TabIndex = 1;
this.pauseButtonXP.Text = "Пауза";
this.toolTip.SetToolTip(this.pauseButtonXP, "Постановка/зняття
процесу обробки з паузи");
this.pauseButtonXP.Click += new
System.EventHandler(this.pauseButtonXP_Click);
//
// closingTimer
//

```

```

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

Кафедра _ КБПЗ _ 2022 рік

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'єм:";
        //
        // 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 = "МАГІСТЕРСЬКА 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 = "Дослідження та програмна реалізація системи
забезпечення цілісності даних у хмарних сервісах";
//
// 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 = "";
//
// 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 Мицак Ілля Миколайович";
//
// 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. KH-21M-1,4";
//
// 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: Помазан Л.В.";
//
// 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 2022";
//
// 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;
}
}

```