Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи структурного кодування даних у комп’ютерних мережах автоматизованих систем управління”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22МЗ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Скрипник Д.А.
«____» _____ 2023 р.

Керівник проекту
доктор філософії (PhD)
_____ Дрєєва Г.М.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

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

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

Скрипнику Дмитру Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління*

2. Керівник роботи *Дрєєва Ганна Миколаївна, доктор філософії (PhD)*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 36-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління*

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

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

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 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	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Скрипник Д.А. Дослідження та програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Метою розробки є дослідження та програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Об'єктом дослідження є процес структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Предметом дослідження є методи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Методи дослідження базуються на методах теорії кодування, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

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'ютерна інженерія, структурне кодування

ABSTRACT

Skrypnyk D.A. Research and software implementation of the system of structural coding of data in computer networks of automated control systems. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software was developed, which is intended for the system of structural coding of data in computer networks of automated control systems.

The purpose of the development is the research and software implementation of the system of structural coding of data in computer networks of automated control systems.

The object of research is the process of structural coding of data in computer networks of automated control systems.

The subject of research is methods of structural coding of data in computer networks of automated control systems.

Research methods are based on coding theory methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the system of structural coding of data in computer networks of automated control systems.

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 engineering, structural coding

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АСУ – Автоматизовані системи управління

ВСТУП

Актуальність теми. Сучасний етап розвитку суспільства характеризується зростаючою роллю автоматизованих систем управління в інформаційній сфері – сукупністю інформації, інформаційної інфраструктури, суб'єктів, що здійснюють формування, поширення, обробку і використання інформації, а також системи регулювання та прийняття рішень у суспільних відносинах, що виникають при цьому. Автоматизовані системи управління є системно утворюючим чинником життя суспільства, що активно впливають на стан політичної, економічної, оборонної й інших складових безпеки держави.

Один з напрямків поліпшення якості управлінських, експлуатаційних і виробничо-господарських процесів у різних галузях діяльності людини полягає в підвищенні ефективності функціонування автоматизованих систем управління (АСУ). В першу чергу це пов’язано із взаємодією АСУ та груп територіально рознесених космічних апаратів й обробкою інформації, отриманої від засобів аерокосмічного моніторингу, з метою забезпеченням ефективної роботи різних міністерств, відомств, забезпеченням безпеки й ефективності функціонування транспортної, нафтогазової й енергетичної галузей. Для забезпечення виконання зазначених напрямків сучасні АСУ містять у собі підсистеми збору, передачі, обробки, зберігання, аналізу, видачі й відображення інформації.

Ґрунтуючись на аналізі вимог до АСУ визначено, що однією із пріоритетних характеристик її функціонування є своєчасна обробка, підвищення ступеню повноти і якості інформації, доведення необхідної кількості корисних даних із заданим ступенем вірогідності до споживача. Це призводить до того, що швидкість зростання обсягів інформаційних потоків на декілька порядків перевищуватиме темпи збільшення швидкості передачі даних каналами зв’язку. Зазначене є однією з основних проблемних сторін функціонування автоматизованих систем обробки інформації й управління.

Таким чином, очевидним є протиріччя між обсягами інформаційних потоків, що необхідно передавати з малою похибкою у реальному масштабі часу, та обмеженими технічними можливостями існуючих інформаційно-комунікаційних систем та мереж АСУ. Економічно найвигіднішим напрямком вирішення цього протиріччя є компактне подання корисної інформації, шляхом використання оптимізованих алгоритмів стиснення інформаційних потоків та неспотворена передача корисних даних у зазначені часові терміни. Існуючі технології компактного подання даних не забезпечують необхідного ступеня стиснення інформаційних потоків, отриманих від різних джерел інформації, що потрібно для своєчасного доведення інформації та ефективного прийняття рішень в АСУ.

Для підвищення ступеня стиснення й збереження заданої вірогідності передачі корисної інформації, створювані методи компактного подання даних, повинні задовольняти наступним вимогам: організовувати процес стиснення даних на основі усунення структурної надмірності; враховувати в процесі стиснення структурні закономірності одночасно за декількома ознаками; обробляти двійкові дані з метою універсалізації подання інформаційних потоків від різних джерел.

Вибір структурних ознак повинен бути обумовлений: можливістю вилучення інформаційної структурної надмірності без втрати якості; кількісною мірою характеристик із врахуванням закономірностей структури; зменшенням, або повним виключенням залежності щодо знань апріорної інформації для реалізації процесу кодування. Однак, методи стиснення на основі структурного кодування у двійковому структурному просторі відсутні.

Таким чином, магістерська робота присвячена підвищенню ефективності функціонування АСУ на основі новітніх методів стиснення інформаційних потоків.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

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

– Огляд існуючих систем структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

– Дослідження системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

– Програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Об'єктом дослідження є процес структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Предметом дослідження є методи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Методи дослідження базуються на методах теорії кодування, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

– Розроблено вітчизняний продукт структурного кодування даних у комп'ютерних мережах автоматизованих систем управління, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

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 LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи структурного кодування даних у комп’ютерних мережах автоматизованих систем управління, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації методів структурного кодування даних у комп’ютерних мережах автоматизованих систем управління. Це кодування призначено для стиснення даних, які передаються по комп’ютерній мережі.

Характерною особливістю більшості типів даних є їх надлишковість. Ступінь надлишковості даних залежить від типу даних. Наприклад, для відеоданих ступінь надлишковості в декілька разів більша ніж для графічних даних, а ступінь надлишковості графічних даних, у свою чергу, більша за ступінь надлишковості текстових даних. Іншим фактором, що впливає на ступінь надлишковості є прийнята система кодування. Прикладом систем кодування можуть бути звичайні мови спілкування, які є ні чим іншим, як системами кодування понять та ідей для висловлення думок. Так, встановлено, що кодування текстових даних за допомогою засобів української мови дає в середньому надлишковість на 20-25% більшу ніж кодування аналогічних даних засобами англійської мови.

Для людини надлишковість даних часто пов'язана з якістю інформації, оскільки надлишковість, як правило, покращує зрозумілість та сприйняття інформації. Однак, коли мова йде про зберігання та передачу інформації засобами комп'ютерної техніки, то надлишковість відіграє негативну роль, оскільки вона приводить до зростання вартості зберігання та передачі інформації. Особливо актуальною є ця проблема у випадку необхідності обробки величезних обсягів інформації при незначних об'ємах носіїв даних. У зв'язку з цим постійно виникає проблема позбавлення надлишковості або стиснення даних. Коли методи стиснення даних застосовуються до готових файлів, то часто замість терміну "стиснення даних" вживають термін "архівування даних", стиснений варіант

даних називають архівом, а програмні засоби, що реалізують методи стиснення називаються архіваторами.

В залежності від того, в якому об'єкті розміщені дані, що підлягають стисненню розрізняють:

1. Стиснення (архівування) файлів: використовується для зменшення розмірів файлів при підготовці їх до передавання каналами зв'язку або до транспортування на зовнішніх носіях малої ємності.

2. Стиснення (архівування) папок: використовується як засіб зменшення обсягу папок перед довготерміновим зберіганням, наприклад, при резервному копіюванні.

3. Стиснення (ущільнення) дисків: використовується для підвищення ефективності використання дискового простору шляхом стиснення даних при записі їх на носії інформації (як правило, засобами операційної системи).

Існує багато практичних алгоритмів стиснення даних, але всі вони базуються на трьох теоретичних способах зменшення надлишковості даних. Перший спосіб полягає в зміні вмісту даних, другий – у зміні структури даних, а третій – в одночасній зміні як структури, так і вмісту даних.

Якщо при стисненні даних відбувається зміна їх вмісту, то метод стиснення є незворотнім, тобто при відновленні (розархівуванні) даних з архіву не відбувається повне відновлення інформації. Такі методи часто називаються методами стиснення з регульованими втратами інформації. Зрозуміло, що ці методи можна застосовувати тільки для таких типів даних, для яких втрата частини вмісту не приводить до суттєвого спотворення інформації. До таких типів даних відносяться відео- та аудіодані, а також графічні дані. Методи стиснення з регульованими втратами інформації забезпечують значно більший ступінь стиснення, але їх не можна застосовувати до текстових даних. Прикладами форматів стиснення з втратами інформації можуть бути: JPEG (Joint Photographic Experts Group) для графічних даних; MPG – для для відеоданих; MP3 – для аудіоданих.

Якщо при стисненні даних відбувається тільки зміна структури даних, то метод стиснення є зворотнім. У цьому випадкові з архіву можна відновити інформацію повністю. Зворотні методи стиснення можна застосовувати до будь-яких типів даних, але вони дають менший ступінь стиснення у порівнянні з незворотними методами стиснення. Приклади форматів стиснення без втрати інформації: GIF (Graphics Interchange Format), TIFF (Tagged Image File Format) – для графічних даних; AVI – для відеоданих; ZIP, ARJ, RAR, CAB, LH – для довільних типів даних. Існує багато різних практичних методів стиснення без втрати інформації, які, як правило, мають різну ефективність для різних типів даних та різних обсягів. Однак, в основі цих методів лежать три теоретичних алгоритми:

– алгоритм RLE (Run Length Encoding);

– алгоритми групи KWE(KeyWord Encoding);

– алгоритм Хафмана.

1.2 Область застосування

Областю застосування є комп'ютерні мережі автоматизованих систем управління. Проведемо аналіз сучасного стану АСУ, приведемо їх загальну класифікацію та характеристики. Встановлені залежності ефективності функціонування АСУ від процесу обробки й передачі даних. Визначені напрямки підвищення якості управлінських та виробничо-господарських процесів за рахунок вдосконалення функціональних можливостей та технічних характеристик автоматизованих систем. До основних характеристик АСУ належать:

– надійність роботи;

– забезпечення заданого ступеня вірогідності інформації;

– швидкодія обчислювальних процесів;

– швидкість передачі (розподілу) інформації.

Були виявлені протиріччя між обсягами інформаційних потоків, що необхідно передавати з як завгодно малою похибкою у реальному масштабі часу, та обмеженими технічними можливостями існуючих інформаційно-комунікаційних систем та мереж АСУ.

Одним з ефективних напрямків зниження обсягів даних без зменшення ступеня вірогідності, є кодування джерела повідомлення на основі двоознакового структурного подання даних.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

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

Movavi Video

Якою програмою стискати відео? Спеціально для новачків раджу «Відео Конвертер» від компанії Movavi.

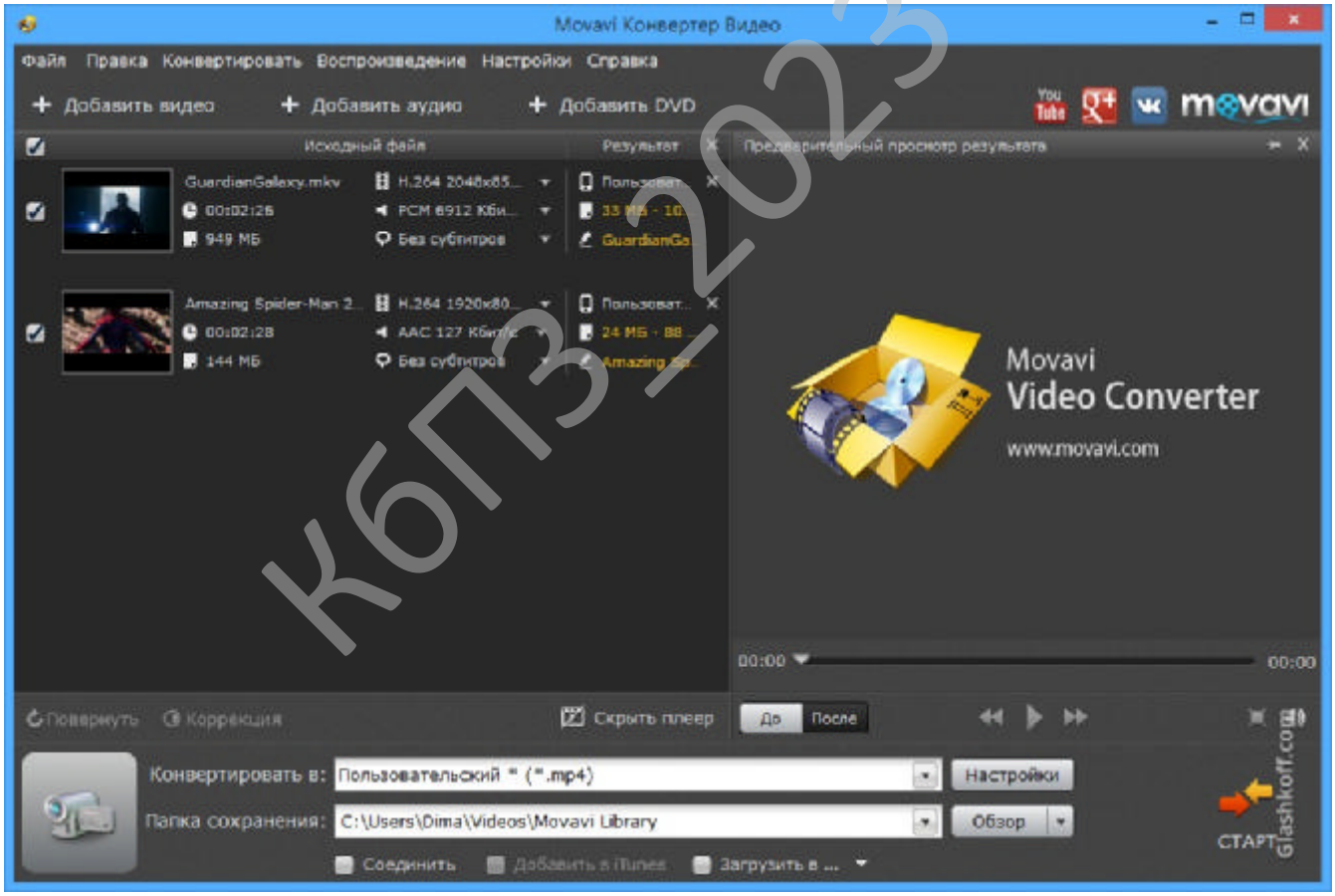

Рисунок 2.1 – Інтерфейс користувача Movavi

Це, мабуть, єдиний відеоконвертер, що володіє відразу декількома корисними перевагами:

– простий у використанні;

– може конвертувати файли з відео;

– може зберегти диск Video-DVD у файл відео;

– має безліч готових налаштувань, що підходять для більшості випадків (конвертація у формат, «перетравлюваний» телефоном, планшетом, плеєром і так далі);

– працює без помилок, має російськомовну технічну підтримку;

– конвертує відео, використовуючи всі доступні ресурси комп'ютера. Автори хвастаються режимом SuperSpeed, завдяки якому програма обробляє відео швидше, ніж її аналоги.

Мінус тільки один – конвертер платний. Без реєстрації дозволяє конвертувати тільки половину відео, ще й накладає «водяний знак» на відео. Коли мова заходить про простоту використання, безпроблемність, швидкість роботи, причому все це одночасно, безкоштовних рішень немає.

AVS Video Converter

Якщо використовується пробна версія, то при конвертації програма повідомить вас, що в центрі відео буде вставлений баннер з рекламою. Проведу невеликий огляд даної програми щоб було ясно. Чим гарна дана програма, так тим, що в програмі є велика кількість розширень виводу, обробленого відео, серед яких:

– DVD video.

– MPEG.

– MP4.

– Blu-Ray.

– Відео для YouTube.

– Формати відео для різних мобільних пристроїв.

З налаштуваннями програми звичайно прийдеться повозитися, щоб вибрати найбільш підходящий варіант і при цьому зменшити розмір кінцевого продукту. Усе буде залежати від частоти кадрів і інших налаштувань. Безсумнівним плюсом даної програми є наявність убудованого відеоредактора, у якому ви можете редагувати своє відео, застосовувати різні фільтри й додавати графіку.

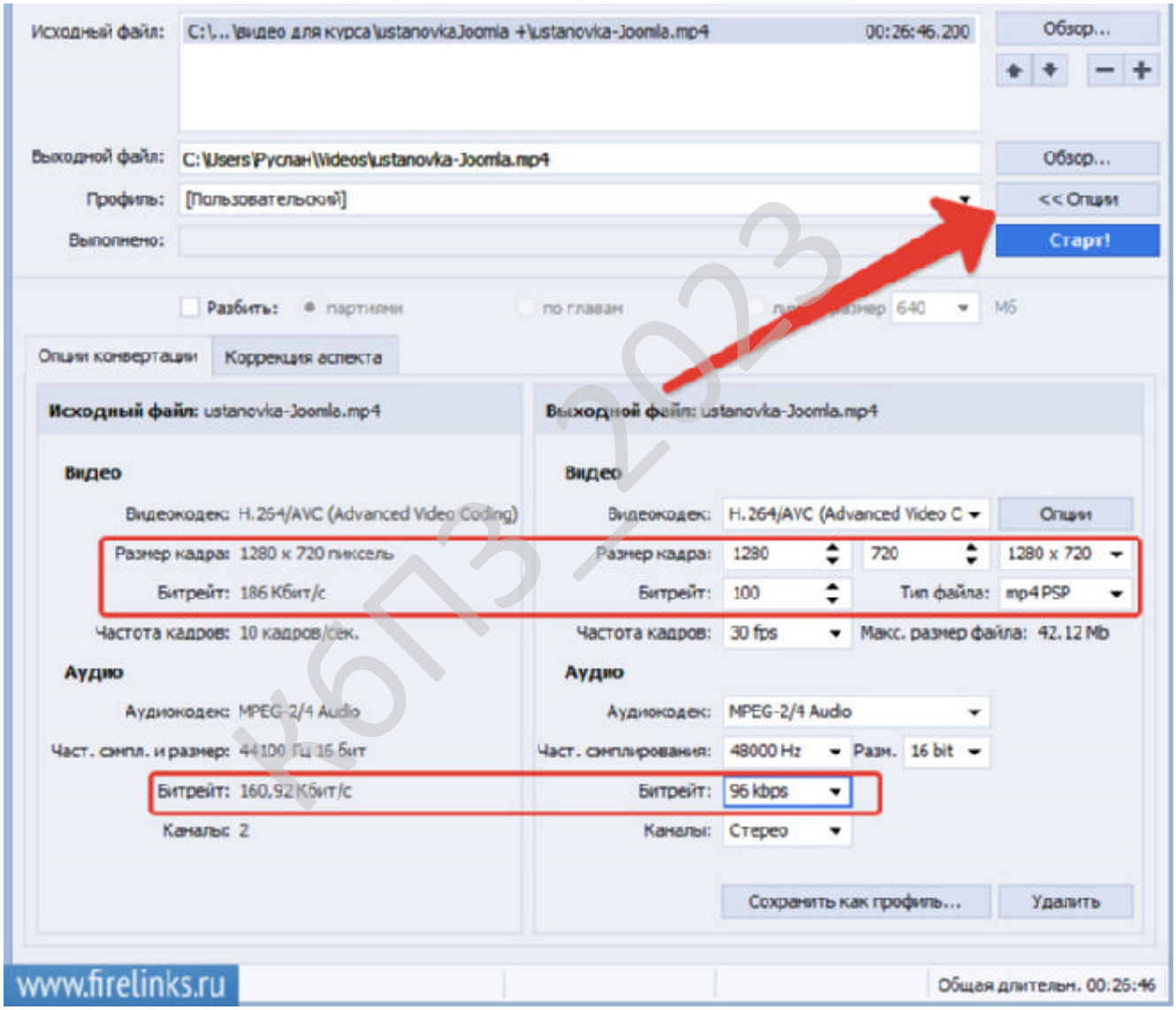

Рисунок 2.2 – Інтерфейс користувача AVS Video Converter

Handbrake

Сама програма повністю безкоштовна, але конвертує тільки формат MP4. Особисто я використовую саме цей формат дуже часто. Ви можете не повірити, але навіть при стандартних налаштуваннях розмір відео зменшується в 5-6 разів, що саме по собі дуже круто. На сьогоднішній день я поки не знайшов програми краще, ніж ця. Причому якість відео в незалежності від його тривалості зберігається як у вихідного.

Принцип роботи досить простий і не вимагає докладного вивчення. Завантажуємо, установлюємо й працюємо. Не забудьте обновити або встановити останню версію Microsoft .NET Framework 4, скачати можна на офіційному сайті Microsoft. Для початку потрібно вибрати наше відео у форматі MP4 і вказати кінцеву папку виводу.

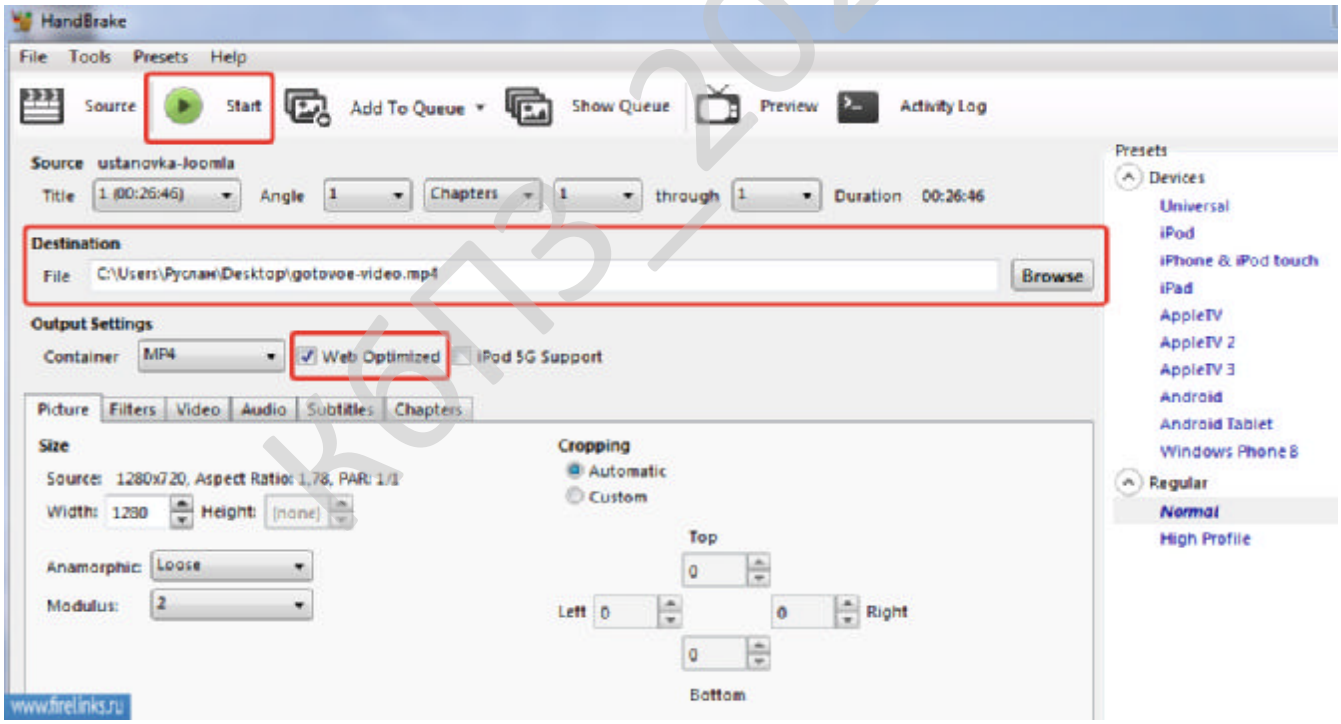

Рисунок 2.3 – Інтерфейс користувача Handbrake

Я записував відео загальною тривалістю більше 2-х годин і, якби не це конвертер загальний розмір у мене виходив близько 9 Гб, а так після стиску розмір став усього 1 Гб.

Відеомайстер

Відеомайстер від компанії AMS Software – досить зручна й широко відома на ринку програма, за допомогою якої можна легко перетворити будь-яку кількість відеофайлів у те або інше розширення. Утиліта має російськомовний інтерфейс і дозволяє працювати як з популярними, так і з рідкими форматами: AVI, MP4, MPEG, 3GP, 3G2, WMV, MOV, VOB, MKV, FLV, MPG і т.д.

Дана програма для стиску відео розроблена з обліком основних користувальницьких вимог. Саме тому вона наочна й неймовірно проста у використанні, на відміну від своїх численних конкурентів.

Рисунок 2.4 – Інтерфейс користувача Відеомайстер

Усе, що необхідно зробити – це вибрати конкретний відеоролик для конвертування, завантажити його в програму, визначитися з форматом і запустити процес конвертації. Додавати можна не тільки окремі файли, але й цілі папки з відео. Додаток підтримує стандартні й високоякісні відео розширення. Крім усього іншого, ви можете настроїти тип кодеку, кращий бітрейт і інші характеристики конвертованого файлу. До речі, конвертер підтримує функцію автоматичного вимикання комп'ютера ПК після закінчення роботи.

Ще одна корисна особливість утиліти полягає в тім, що «Відеомайстер» здатний конвертувати відеоролики для різних гаджетів. Таким чином, якщо у вас є iPhone, iPod або інший сучасний девайс, ви можете захитати на свій пристрій потрібний вам фільм або кліп. У вашім розпорядженні більше 250 передвстановок практично для будь-якої моделі смартфона, портативного плеєра або ігрової консолі.

Також програма для стиску відео дозволяє завантажувати файли на сайт або блог. Можна опублікувати ролик на YouTube або Вконтакті без прямого звертання до самому хостингу, тому що конвертер генерує FLV і SWF відео із уже готовим медіа-плеєром. Зовнішній вигляд плеєра й необхідні параметри відеозапису налаштовуються завчасно.

Крім перерахованого вище, додаток оснащений можливістю обробки відео в убудованому редакторі. Використовуючи багату колекцію фільтрів для обробки, кліпи можна кадрувати, прикрашати їх усілякими спецеффектами, поліпшувати якість зображення, додавати написи й логотипи, витягати звук і багато чого, багато чого іншого.

«Відеомайстер» – це не просто універсальна програма для стиску відео, за допомогою якої будь-який користувач може зменшити розмір відеозапису, що цікавить. Програма також допоможе конвертувати відео в DVD, обрізати або з'єднати певні фрагменти. Завдяки широкому діапазону можливостей і зручної навігації, цей конвертер стане вашим незамінним помічником при роботі з відео будь-якого формату.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

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

Як об'єктно-орієнтована мова, Visual C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод Main – крапку входу додатка – інкапсулюється у визначення класів. Клас може успадковувати безпосередньо з одного родового класу, але може реалізовувати

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

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

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

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

– Атрибути з декларативними метаданими про типи під час виконання.

– Вбудовані коментарі XML-документації.

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

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

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

Архітектура платформи.NET Framework

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

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

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

Взаємодія між мовами є ключовою особливістю.NET Framework. Оскільки код IL, створюваний компілятором Visual C# відповідає специфікації CTS, код IL на основі Visual C# може взаємодіяти з кодом, створюваним версіями мов Visual Basic, Visual C++, Visual J# платформи.NET Framework і ще більш ніж 20 CTS-сумісних мов. В одному складанні може бути кілька модулів, написаних на

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

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;

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Даний розділ присвячений розробці методів одноознакового структурного кодування двійкових даних за кількістю серій одиниць з метою підвищення ступеню стиснення інформаційних потоків в АСУ. Визначимо додатковий облік обмежень на кількість серій одиничних елементів у двійкових послідовностях чисел забезпечує збільшення ступеня стиснення інформаційного потоку. Проведемо оцінку ступеня стиснення двійкових даних в комп'ютерних мережах АСУ на основі розробленого кодування.

Встановлено, що суть кодування полягає у формуванні коду-номера $N(m,\Lambda,\vartheta)_j$ всієї двійкової послідовності із заданим значенням структурної ознаки. Визначено, що структурними ознаками є: вектор S заборон появи на певній позиції одиничного елемента ($\overline{S}=\{s_i\}, i=\overline{1,m}$; s_i – ознака заборони появи на i-й позиції одиничного елемента; якщо $s_i=0$, те на i-й позиції заборонена поява одиниці й навпаки); кількість серій одиниць ϑ у двійковій послідовності. Кодування з урахуванням заборон на позиції одиниць відповідає структурному представленню двійкових даних. Двійкова послідовність чисел розглядається, як структурне число елементи якого набувають значення $\{0; 1\}$.

За результатами досліджень, отримано систему виразів одноознакового структурного кодування двійкових даних у структурному просторі, що забезпечує формування коду-номера відповідно двійковому структурному числу, елементи якого задовольняють заданій структурній ознаці (система виразів, що визначає номер заданої двійкової послідовності в безлічі одноознакових структурних чисел).

Встановлено, що безлічі двійкових послідовностей у структурному просторі за кількістю серій одиниць $V(m,\Lambda,\vartheta)$ відповідає пронумерована

множина двійкових послідовностей (1, 2), що містять задану кількість серій одиниць та не проходять через позиції із забороною одиниць.

$$V(m,\Lambda,\vartheta)=\sum_{k=1}^{K}V(\Theta^{(k)})=\sum_{k=1}^{K}\prod_{z=1}^{Z}V(\vartheta_z^{(k)},\Theta^{(k)}) \tag{3.1}$$

$$V(\vartheta_z^{(k)},\Theta^{(k)})=\begin{pmatrix} m_z+1 \\ 2\vartheta_z^{(k)} \end{pmatrix}=\frac{(m_z+1)!}{(2\vartheta_z^{(k)})!\ (m_z+1-2\vartheta_z^{(k)})!}, \tag{3.2}$$

де

$\vartheta_z^{(k)}$ – значення числа серій одиниць для **z**-ї припустимої зони двійкової послідовності A;

$\Theta^{(k)}$ – вектор, елементами якого є k-та комбінація кількостей серій одиниць $\vartheta_z^{(k)}$ у припустимих зонах $\Theta^{(k)}=\{\vartheta_1^{(k)},...,\vartheta_z^{(k)},...,\vartheta_Z^{(k)}\}$, $k=\overline{1,\ K}$;

Z – кількість припустимих зон у двійковій послідовності;

m_z – кількість двійкових елементів в z-й припустимій зоні;

$V(\vartheta_z^{(k)},\Theta^{(k)})$ – кількість припустимих двійкових послідовностей, отриманих для z-ї припустимої зони за кількістю серій одиниць, рівній $\vartheta_z^{(k)}$ для вектора $\Theta^{(k)}$.

Показано, що величина $V(\vartheta_z^{(k)},\Theta^{(k)})$ дорівнює кількості двійкових підпослідовностей, що утворюють відповідну z-у зону, у яких число серій одиниць дорівнює $\vartheta_z^{(k)}$. У зв'язку із цим z-м елементом безлічі $\Psi(\Theta^{(k)})$ є величина коду-номера $N(\vartheta_z^{(k)},\Theta^{(k)})$, для якого виконується нерівність

$$N(\vartheta_z^{(k)},\Theta^{(k)})\le V(\vartheta_z^{(k)},\Theta^{(k)})-1. \tag{3.3}$$

Визначено, що кодування двійкових структурних чисел за кількістю серій одиниць є система виразів, що дозволяє визначити код-номер $N(m,\Lambda,\vartheta)_j$ оброблюваної двійкової послідовності $A^{(j)}=\{a_{ij}\}_{i=\overline{1,m}}$ з виявленими обмеженнями на позиції одиниць $\Lambda=\{\lambda_i\}_{i=\overline{1,m}}$ та на кількість серій одиниць ϑ у безлічі двійкових структурних чисел, що містять задану кількість серій одиниць.

Встановлено, що одноознакове подання двійкових даних у структурному просторі є взаємооднозначним, а саме: за заданим кодом номером $N(m,\Lambda,\vartheta)_j$ можна відновити тільки одну вихідну двійкову послідовність $A^{(j)}=\{a_{ij}\}_{i=\overline{1,m}}$

відповідну даній кодовій конструкції без внесення похибки, тобто

$$N(m,\Lambda,\vartheta)_j = \varphi_k\left(A^{(j)}\right) \text{ і } A^{(j)^*} = \varphi_d\left(N(m,\Lambda,\vartheta)_j\right) \tag{3.4}$$

де

$A^{(j)} = \{a_{izj}\}_{i=\overline{1,m}}$, $A^{(j)^*} = \{a^{\bullet}_{izj}\}_{i=\overline{1,m}}$; $a^{\bullet}_{izj} = a_{izj}$, $i=\overline{1,m}$; $z=\overline{1,Z}$, φ_k, φ_d – відповідно оператори кодування й декодування двійкових структурних чисел з урахуванням обмеженої кількості серій одиниць;

$A^{(j)}$, $A^{(j)^*}$ – відповідно вихідна й відновлена двійкові послідовності;

a_{izj}, $a^{\bullet}_{izj}$ – i,z,j-е елементи відповідно до вихідної і відновленої двійкових послідовностей, що належать **z**-й припустимій зоні.

Розроблено систему правил та алгоритмів відновлення двійкових структурних чисел з обмеженою кількістю серій одиниць.

Встановлено, що вихідну послідовність $A^{(j)} = \{a_{izj}\}_{i=\overline{1,m}}$, яка задовольняє обмеженням

$$0 \le a_{ij} \le s_i;\ s_i = \max_{1\le j\le n}\{a_{ij}\};\ a_{ij};\ s_i \in \{0;1\};\ \vartheta = \sum_{z=1}^{Z}\vartheta_z^{(k)} \text{ і } 0 \le \vartheta_z^{(k)} \le \min\{\vartheta;[\frac{m_z+1}{2}]\}$$

можна відновити без внесення похибки на основі значень коду-номера $N(m,\Lambda,\vartheta)_j$, обмежень на позиції одиниць $\Lambda = \{\lambda_i\}_{i=\overline{1,m}}$ і на кількість серій одиниць ϑ, тобто виконується умова $A_\alpha^{(u)} = A_\alpha^{(j)}$, де $A_\alpha^{(u)}$ *і* $A_\alpha^{(j)}$ вихідна й відновлена двійкові послідовності, $\alpha = \overline{1,\varsigma}$.

Встановлено, що мінімальне значення коефіцієнта стиснення $k_{\min}$ у результаті одноознакового кодування двійкових даних у структурному просторі дорівнює:

$$k_{\min} = \frac{\ell og_2 m}{\ell og_2 V(m,\Lambda,\vartheta)}\ , \tag{3.5}$$

де $V(m,\Lambda,\vartheta)$ – кількість двійкових чисел довжиною m елементів з кількістю серій одиниць ϑ у структурному просторі із заданим вектором обмежень Λ.

Встановлено, що величина $k_{\min}$ є верхньою межею для значення коефіцієнта стиснення k_{cm}:

$$k_{cm} = \frac{\ell og_2 m}{\ell og_2 N(m, \Lambda, \vartheta)} < k_{\min} \quad . \tag{3.6}$$

За рахунок розробленого кодування забезпечується додаткове підвищення ступеня стиснення щодо структурного кодування й структурного кодування за кількістю серій при цьому, виконуються відповідні нерівності:

$$\ell og_2 V(m, \Lambda, \vartheta) \le \ell og_2 V(m, \Lambda), \tag{3.7}$$

$$\ell og_2 V(m, \Lambda, \vartheta) \le \ell og_2 V(m, \vartheta), \tag{3.8}$$

де $\ell og_2 V(m, \Lambda)$ й $\ell og_2 V(m, \vartheta)$ – кількість розрядів, що відведено для подання відповідно структурного й структурного числа з обмеженою кількістю серій одиниць та складені із m двійкових елементів кожне.

На підставі математичного моделювання й проведеного аналізу побудовані графіки залежностей величини мінімального коефіцієнта стиснення k_{cm} від m_z й $\vartheta_z^{(\xi)}$ (рисунок 3.1).

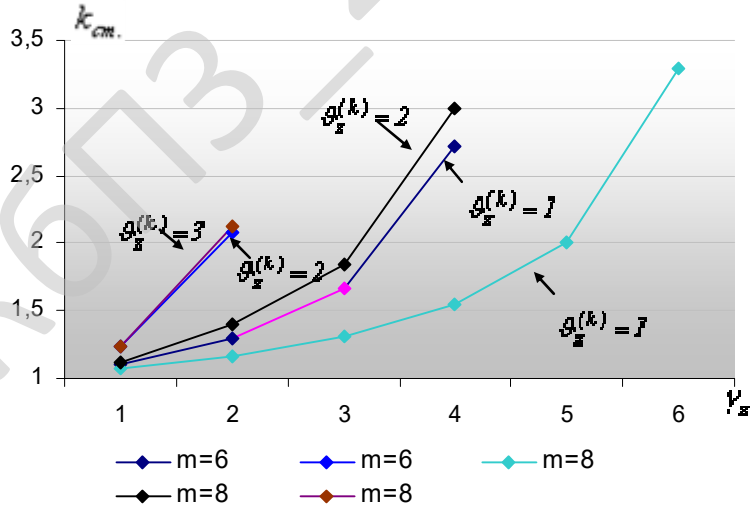

Рисунок 3.1 – Графіки залежності коефіцієнта стиснення $k_{cm.}$ від довжини двійкової послідовності m_z, числа серій одиниць $\vartheta_z^{(k)}$ і γ_z- кількість заборонених позицій одиниць в z-й зоні

З аналізу отриманих результатів випливає таке.

По-перше, максимальне значення додаткового коефіцієнта стиснення k_{cm} забезпечується зі збільшенням кількості позицій із забороненою появою одиниць.

Для $m_z=8$ й $\vartheta_z^{(k)}=1$ при переході від $\gamma_z=1$ до $\gamma_z=6$ величина k_{cm} збільшується в 3,1 рази, для $m_z=8$ й $\vartheta_z^{(k)}=2$ при переході від $\gamma_z=1$ до $\gamma_z=4$ величина k_{cm} збільшується в 2,67 рази, а для $m_z=6$ й $\vartheta_z^{(k)}=1$ при переході від $\gamma_z=1$ до $\gamma_z=4$ величина k_{cm} збільшується в 2,47 рази.

По-друге, для фіксованих значень кількості заборонених позицій одиниць в z-й зоні найбільші значення величини k_{cm} досягаються із збільшенням кількості серій одиниць.

Для $m_z=8$ й $\gamma_z=2$ зі збільшенням кількості серій одиниць від $\vartheta_z^{(k)}=1$ до $\vartheta_z^{(k)}=3$ величина k_{cm} збільшується в 1,98 рази. Аналіз отриманих результатів показує, що величина k_{cm} для $\vartheta_z^{(k)}=1$, $\gamma_z=1$ збільшується в 1,15 разів при зменшенні довжини припустимої зони від $m_z=8$ до $m_z=4$ (рисунок 3.2).

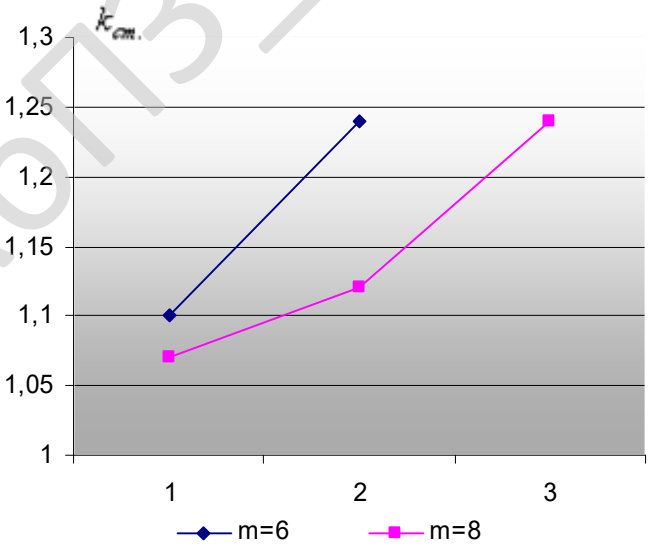

Рисунок 3.2. – Графіки залежності коефіцієнта стиснення $k_{cm.}$ від m_z і $\vartheta_z^{(k)}=1$ для $\gamma_z=1$

Отже, за результатами досліджень сформовано метод одноознакового структурного кодування двійкових даних по кількості серій одиниць.

Встановлено, що розроблена система правил є взаємооднозначною для процедур кодування і декодування інформаційних потоків в АСУ.

Встановлено, що додатковий облік обмежень на кількість серій одиничних елементів у двійкових послідовностях чисел забезпечує збільшення ступеня стиснення даних в АСУ без втрат інформації.

3.2 Розробка структурної схеми

У даному розділі наведено розробку методів двоознакового структурного кодування двійкових даних з метою додаткового збільшення ступеня стиснення інформації в АСУ.

Структурними ознаками для двійкових послідовностей виступають: кількість серій одиниць у кожній припустимій зоні й межі припустимих зон.

Встановлено, що додаткове збільшення ступеня стиснення двійкових даних без внесення похибки в АСУ досягається в результаті обліку припустимих комбінацій тільки для одного вектора обмежень на число серій одиниць у припустимих зонах.

Виведемо аналітичні вирази, що дозволяють формувати код-номер для двійкових послідовностей, що задовольняють одночасно двом структурним ознакам.

Встановимо, що двоознакове подання двійкових структурних чисел є взаємооднозначним, тобто відновлення вихідних даних здійснюється без внесення похибки в інформаційний потік АСУ.

Розробимо метод відновлення двійкових даних на основі двоознакового структурного декодування. Покажемо, що двійкові послідовності, що задовольняють системі структурних обмежень на число серій одиниць у припустимих зонах є двоознаковими двійковими структурними числами.

Множина, складена із двійкових послідовностей, що одночасно задовольняє структурним обмеженням:

– на позиції із припустимою появою одиничних елементів, що задається вектором обмежень на діапазон значень оброблюваних елементів Λ;

– сумарне число серій одиниць ϑ у всій оброблюваній послідовності (перша структурна ознака);

– число серій одиниць $\vartheta_{z,j}$ у кожній припустимій зоні (друга структурна ознака – задається вектором $\Theta^{(k)}$ значень величин $\vartheta_z^{(k)}$) – визначена , як множина двоознакових двійкових структурних чисел.

Встановлено, що двоознаковою структурною нумерацією даних у двійковому структурному просторі є процес обчислення порядкового номера відповідно до двоознакового двійкового структурного числа в припустимій безлічі, що складається з двійкових послідовностей задовольняючій системі обмежень:

$$\begin{cases} 0 \le a_{ij} \le s_i,\ i=\overline{1,m}; \\ \vartheta = \sum_{z=1}^{Z} \vartheta_z^{(k)}; \\ \vartheta_z^{(k)} = \vartheta_{z,j},\ z=\overline{1,Z}, \end{cases} \tag{3.9}$$

де

s_i – обмеження на позиції, що допускають появу одиничних елементів;

$\vartheta_z^{(k)}$ – припустиме значення обмеження на число серій одиниць в z-й зоні;

$\vartheta_{z,j}$ – значення числа серій одиниць в z-й зоні, обчислене для конкретної оброблюваної j-ї двійкової послідовності.

На основі аналізу встановлено, що значення коду-номера $N\left(m,\Lambda,\Theta^{(k)}\right)_j$ двоознакового двійкового структурного числа буде меншим у порівнянні зі значенням коду-номера $N\left(m,\Lambda,\vartheta\right)_j$ одноознакового структурного числа $N\left(m,\Lambda,\Theta^{(k)}\right)_j \le N\left(m,\Lambda,\vartheta\right)_j$.

Встановимо взаємооднозначність двоознакового структурного подання двійкових даних у структурному просторі. Для обраного лексикографічного правила нумерації, заданих обмежень на число серій одиниць у кожній припустимій зоні й обмежень на розташування одиничних елементів у двійковій послідовності $A(m,\Theta^{(k)})_j$, можна сформувати тільки один код-номер $N(m,\Lambda,\vartheta)_j$ відповідно до даної послідовності.

Визначено, що для двійкової послідовності $A(m,\Theta^{(k)})$ розглянутої, як двоознакове структурне число із заданими структурними параметрами й обмеженнями, що накладені на них (довжина послідовності m, обмеження Λ на позиції із припустимою появою одиничних елементів та обмеження $\Theta^{(k)}$ на число серій одиниць у припустимих зонах) – можна сформувати тільки один код-номер $N\left(m,\Lambda,\Theta^{(k)}\right)_j$ відповідно до цієї послідовності.

Розроблено систему правил відновлення двійкових двоознакових структурних чисел у вигляді структурної схеми системи (рисунок 3.3). Двійкову послідовність $A(m,\Theta^{(k)})_j=\{a_{izj}\}_{i=\overline{1,m}}$, що задовольняє системі обмежень можна відновити без внесення похибки на основі значень коду-номера $N\left(m,\Lambda,\Theta^{(k)}\right)_j$, з урахуванням відомих значень величин: довжини послідовності m, вектора обмежень на позиції одиниць $\Lambda=\{\lambda_i\}_{i=\overline{1,m}}$ і вектора $\Theta^{(k)}$ обмежень на число серій одиниць у припустимих зонах. На основі визначених переваг двоознакового відновлення відносно одноознакового поелементного відновлення встановлено, що є можливість здійснювати відновлення елементів z-ї припустимої зони незалежно від процесу відновлення попередніх $(z-1)$ зон.

Таким чином, для додаткового підвищення ступеня компактного подання й зниження часу на кодування необхідно на одноознакові структурні двійкові послідовності накладати додаткове обмеження на комбінацію числа серій одиниць у припустимих зонах.

Рисунок 3.3 – Структурна схема системи

Встановлено, що: верхньою межею обсягу безлічі двоознакових чисел є обсяг безлічі одноознакових структурних чисел; кількість розрядів, що відведено на подання коду-номера двоознакового двійкового структурного числа не буде перевищувати кількість розрядів, що витрачаються на подання коду-номера одноознакового двійкового числа в структурному просторі.

Розроблимо методику оцінки ступеня компактного подання двоознакових структурних чисел. Здійснимо порівняльну оцінку характеристик розробленого компактного подання даних з відомими методами. На основі досліджень та математичного моделювання побудуємо графіки залежності величин мінімальних ступенів стиснення.

На основі проведених досліджень встановлено наступне.

По-перше, при фіксованих параметрах $m=8$, $Z=2$, $\vartheta=2$ значення мінімального ступеня стиснення для двоознакового подання збільшується в середньому на 30% при збільшенні кількості заборонених двійкових елементів в інформаційних потоках АСУ (рисунок 3.4).

Коефіцієнти $k(m,\Lambda)_{\min}$, $k(m,\vartheta)_{\min}$, $k(m,\vartheta,\Lambda)_{\min}$, $k(m,\Lambda,\Theta^{(k)})_{\min}$ показані на рисунку 3.4, характеризують стиснення інформаційного потоку на основі структурного кодування, кодування з урахуванням обмеженого числа серій одиниць, одноознакового й двоознакового кодування у двійковому структурному просторі відповідно. Дані коефіцієнти є функції від параметрів: кодування довжини двійкової послідовності m, числа серій одиниць ϑ, кількості припустимих зон Z, кількості заборонених двійкових елементів γ і вектора обмежень $\Theta^{(k)}$ на число серій одиниць у припустимих зонах.

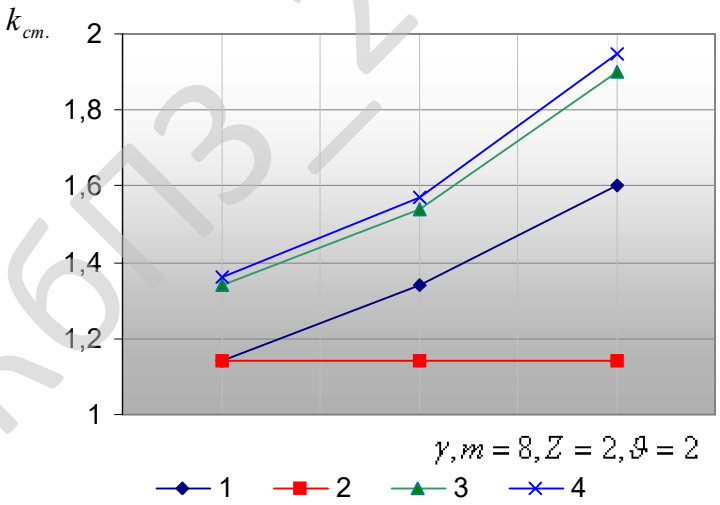

Рисунок 3.4 – Графіки залежності величин $k(m,\Lambda)_{\min}$, $k(m,\vartheta)_{\min}$, $k(m,\vartheta,\Lambda)_{\min}$ і $k(m,\Lambda,\Theta^{(k)})_{\min}$ від γ й $m=8$, $Z=2$, $\vartheta=2$ для 1 – структурного кодування; 2 – кодування з обмеженням на число серій одиниць; 3 – одноознакового структурного кодування; 4 – двоознакового структурного кодування

По-друге, при фіксованих параметрах $m=16$, $\vartheta=2$ і γ мінімальне значення ступеня стиснення для двоознакового структурного кодування збільшується на 20% із збільшенням кількості припустимих зон Z (рисунок 3.5).

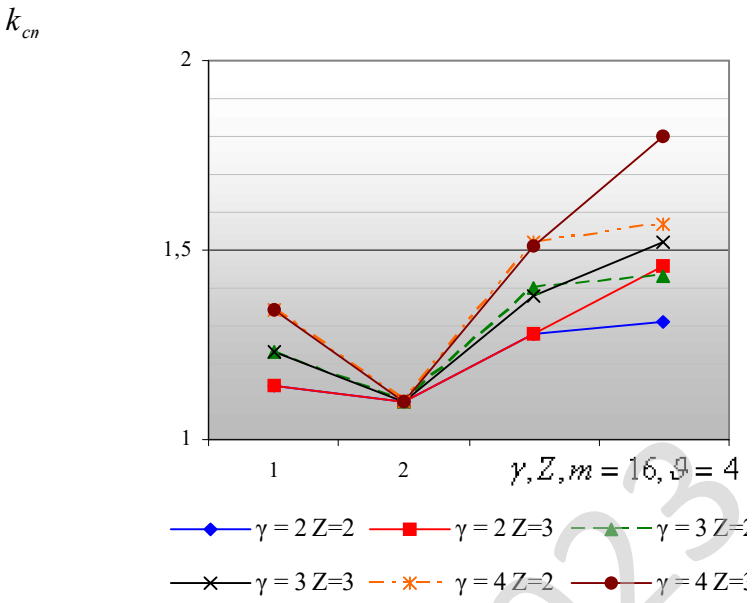

Рисунок 3.5 – Графіки залежності величин $k(m,\Lambda)_{\min}$, $k(m,\vartheta)_{\min}$, $k(m,\vartheta,\Lambda)_{\min}$ і $k(m,\Lambda,\Theta^{(k)})_{\min}$ від γ й Z, $m=16$, $\vartheta=4$ для: 1-структурного кодування; 2- кодування з обмеженням на число серій одиниць; 3-одноознакового структурного кодування; 4- двоознакового структурного кодування

По-третє, у випадку обробки двійкового подання компонент ортогональних перетворень, двоознакове кодування досягає найбільшого значення мінімального ступеня стиснення – 6,25 разів.

При цьому перевага за мінімальним значенням ступеня стиснення для двоознакового кодування щодо структурного й кодування за кількістю серій досягає відповідно 70% і 90% для інформаційних мереж АСУ.

Існує тенденція збільшення мінімального ступеня стиснення для двоознакового структурного подання при збільшенні довжини оброблюваної послідовності.

По-четверте, при обробці на двійковому рівні кодових комбінацій архіваторів і графічних форматів найбільше значення мінімального ступеня стиснення для двоознакового кодування досягає 5 разів.

При цьому виграш за мінімальним значенням ступеня стиснення для двоознакового кодування відносно структурного й кодування за кількістю серій досягає відповідно 2,5 й 4,5 разів.

Існує тенденція зниження значення мінімального ступен стиснення для двоознакового структурного подання при збільшенні кількості припустимих зон у середньому на 10% на одну зону.

За рахунок наявності структурних закономірностей в оброблюваних даних, мінімальне значення ступеня стиснення збільшується відповідно для ортогональних перетворень у 3,5 рази й для кодових комбінацій архіваторів у 2,5 рази (рисунок 3.6.). Таким чином, розроблено методи двоознакового структурного кодування, що забезпечують додаткове підвищення ступеня стиснення даних в комп'ютерних мережах АСУ.

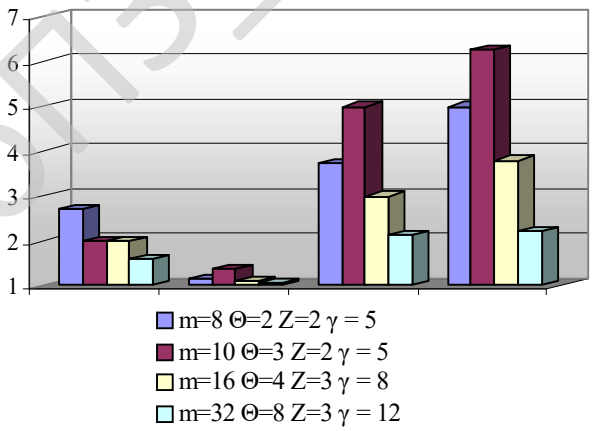

Рисунок 3.6 – Графіки залежності величин $k(m,\Lambda)_{\min}$, $k(m,\vartheta)_{\min}$, $k(m,\vartheta,\Lambda)_{\min}$ і $k(m,\Lambda,\Theta^{(x)})_{\min}$ від γ, Z, m, ϑ для компактного подання компонентів ортогональних перетворень на основі: 1 – структурного кодування; 2 – кодування з обмеженням на число серій одиниць; 3 – одноознакового структурного кодування; 4 – двоознакового структурного кодування

Встановлені залежності для мінімального значення ступеня стиснення двоознакового структурного кодування в двійковому просторі від наявності структурних особливостей в оброблюваних даних залежно від структурних параметрів: довжина двійкової послідовності m, число серій одиниць ϑ, кількість припустимих зон Z, кількість заборонених двійкових елементів γ і компоненти вектора обмежень $\Theta^{(k)}$ щодо числа серій одиниць у припустимих зонах.

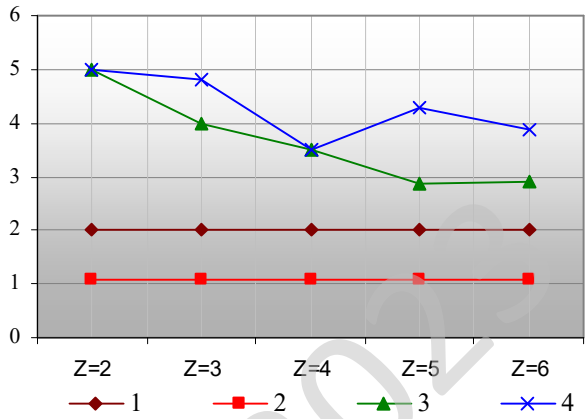

Рисунок 3.7 – Графіки залежності величин $k(m,\Lambda)_{\min}$, $k(m,\vartheta)_{\min}$, $k(m,\vartheta,\Lambda)_{\min}$ і $k(m,\Lambda,\Theta^{(k)})_{\min}$ від Z, $m=32$, $\vartheta=8$ $\gamma=16$ для компактного подання кодових комбінацій архіваторів на основі: 1 – структурного кодування; 2 – кодування з обмеженням на число серій одиниць; 3 – одноознакового структурного кодування; 4 – двоознакового структурного кодування

3.3 Розробка функціональної схеми

У даному розділі розглянуто методи додаткового зниження кількості операцій обчислення двійкових даних з метою мінімізації загального часу на обробку і передачу кодових послідовностей в АСУ баз втрат інформації.

Розробимо методи швидкого кодування й декодування двоознакових структурних чисел. Ці методи базуються на рекурентному знаходженні вагових коефіцієнтів елементів двоознакових чисел у двійковому структурному просторі

на основі позонної обробки вихідних послідовностей. Проведемо оцінку ефективності характеристик розроблених методів відносно існуючих технологій для інформаційних мереж АСУ.

Встановлено, що недоліком двійкового двоознакового подання даних у структурному просторі, є те, що обчислення вагових коефіцієнтів для визначення коду-номера $N\left(m,\Lambda,\Theta^{(k)}\right)_j$ двійкових елементів a_{izj} проводиться на основі обчислення факторіальних виразів. У цьому випадку застосовується велика кількість операцій додавання, множення й ділення.

Розробимо методику зниження кількості операцій обчислення у випадку послідовної обробки, що полягає в організації рекурентного обчислення величин вагових коефіцієнтів. Дана процедура приводить до скорочення надлишкової кількості операцій на кодування.

Розроблено систему вирішальних правил щодо формування коду-номера $N\left(m,\Lambda,\Theta^{(k)}\right)_j$ двійкового двоознакового числа $A^{(j)}=\{a_{ij}\}_{i=\overline{1,m}}$ в структурному просторі $\Lambda=\{\lambda_i\}_{i=\overline{1,m}}$ (з урахуванням виключення надлишкової кількості операцій) на основі рекурентних виразів.

Отримана система виразів забезпечує послідовне поелементне рекурентне кодування двоознакових двійкових структурних чисел і дозволяє проводити обробку (обчислення вагових коефіцієнтів) елементів z-ї зони після обробки елементів $a_{i\gamma j}$ (де $\gamma=\overline{1,\,z-1}$) попередніх зон.

Така незалежна обробка елементів окремих зон відповідає властивості структуризації двоознакових чисел.

Максимальна кількість операцій для рекурентної обробки двоознакових структурних чисел у напрямі, починаючи з молодших розрядів зменшується: в порівнянні з факторіальною обробкою від 5 до 15 разів залежно від довжини оброблюваної послідовності; в порівнянні з рекурентною обробкою в напрямі, починаючи із старших розрядів в середньому в 1,58 разів.

$\mu^{(\partial к)}$, $\mu_1^{(рдк)}$, $\mu_2^{(рдк)}$ оп.

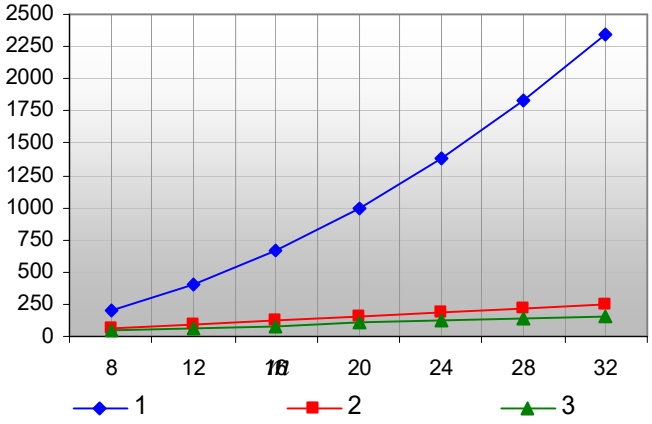

Рисунок 3.8 – Графік залежності кількості операцій $\mu^{(\partial к)}$, $\mu_1^{(рдк)}$ й $\mu_2^{(рдк)}$ від величини m: факторіальна обробка; рекурентна обробка у напрямі молодших розрядів; рекурентна обробка у напрямі старших розрядів

Показано, що послідовність $A_z^{(j)}$ складається з елементів z-ї зони є укрупненим елементом структурного числа. Оскільки за визначенням елементи a_{izj} z-ї зони задовольняють систему обмежень на число серій одиниць у припустимих зонах, то їм відповідає код-номер

$$N(\vartheta_{zj}^{(k)}) = f_{\text{рдк}}(A_z^{(j)}, \vartheta_{zj}^{(k)}, k),\ z=\overline{1, Z},$$

де $f_{\text{рдк}}(\bullet)$ – функціонал послідовного рекурентного двоознакового кодування.

Встановлено, що значення коду-номера $N\left(m, \Lambda, \Theta^{(k)}\right)_j$ формується для послідовності елементів

$$N(\vartheta_{zj}^{(k)}),\ z=\overline{1, Z},$$

$$N\left(m, \Lambda, \Theta^{(k)}\right)_j = \sum_{z=1}^{Z} N(\vartheta_{zj}^{(k)}) \prod_{\phi=z+1}^{Z} V(\vartheta_{\phi j}^{(k)}) \quad . \tag{3.10}$$

Визначено, оскільки величина коду-номера z-ї зони $N(\vartheta_{zj}^{(k)})$ за значенням обмежено зверху величиною $V(\vartheta_{zj}^{(k)}) - 1$, то накопичений добуток $\prod_{\phi=z+1}^{Z} V(\vartheta_{\phi j}^{(k)})$ є його ваговим коефіцієнтом. Це дозволяє організувати позонне рекурентне формування кодів-номерів $N\left(m, \Lambda, \Theta^{(k)}\right)_j$.

Встановлено, що величина $N(\vartheta_{zj}^{(k)})$ визначається, як незалежна від величин $N(\vartheta_{\gamma j}^{(k)})$ інших зон, $\gamma = \overline{1, Z}$, $\gamma \neq z$.

Звідси визначено, що існує можливість організувати паралельне рекурентне двоознакове структурне кодування двійкових даних у структурному просторі.

Розроблено паралельно-рекурентне двоознакове кодування даних у двійковому структурному просторі в основу якого покладена паралельна обробка припустимих зон двійкового двоознакового структурного числа та рекурентна обробка двійкових елементів у межах припустимої зони. Встановлено, що тимчасові витрати для паралельно-рекурентної обробки зменшуються в середньому в Z разів відносно існуючих методів. Розроблене паралельно-рекурентне двоознакове декодування має позонний характер, зворотній запропонованому методу кодування (рисунок 3.9).

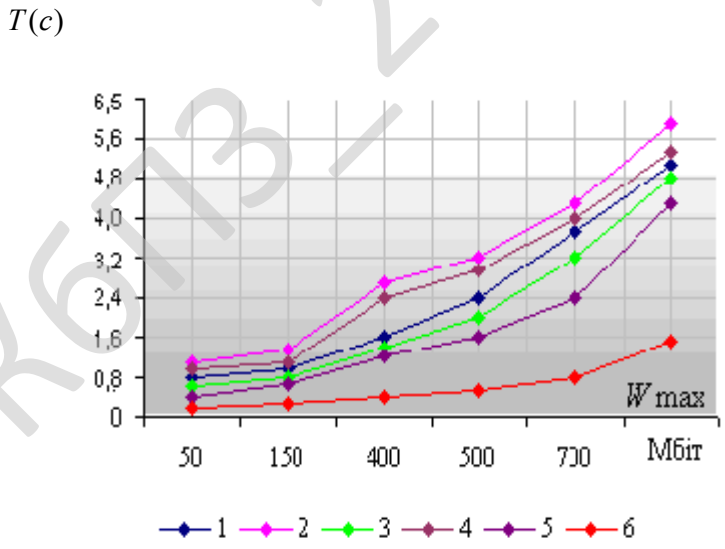

Рисунок 3.9 – Оцінка часу обробки й передачі даних по каналу зв'язку (швидкість-155 Мбіт\с) стиснених різними методами : 1-Алгоритм Хаффмана, 2-LZW, 3-JPEG, 4-Арифметичне кодування, 5-JPEG 2000, 6-Двоознакове структурне кодування

Визначено, що метод послідовного двоознакового деко-дування двійкових даних у структурному просторі, забезпечує скорочення кількості операцій у середньому на 80% (рисунок 3.10).

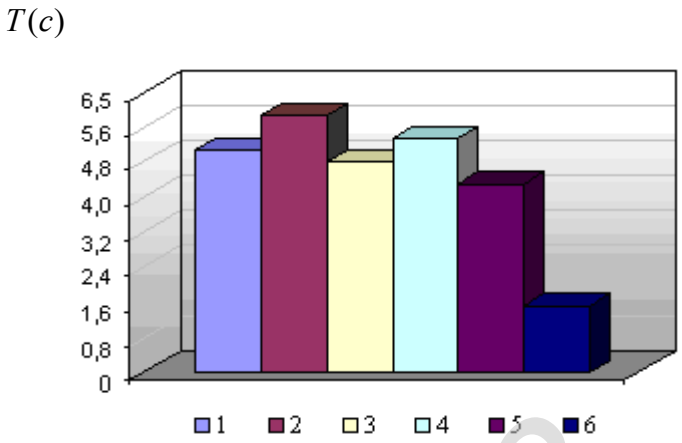

Рисунок 3.10 – Оцінка часу на обробку й передачу даних обсягом W=1 Гбіт по каналу зв'язку (швидкість 155 Мбіт\с), стислих різними методами 1-Алгоритм Хаффмана, 2-LZW, 3-JPEG, 4-Арифметичне кодування, 5- JPEG 2000, 6-Двоознакове структурне кодування

Отже, оскільки дані структурні числа мають властивість зонності, характерну для структурних чисел, існує можливість організовувати паралельно-рекурентне подання елементів для кожної зони, яке не залежить від розрахунку елементів в інших зонах та забезпечує зниження кількості операцій на відновлення вихідних послідовностей і забезпечує скорочення загального часу на обробку та передачу інформації в мережах АСУ.

Розробимо методи оцінок і корекції завадостійкості структурних кодових конструкцій до помилок у каналі передачі даних АСУ. При цьому враховується як особливості моделі появи помилок у кодах при їх передачі по каналу зв'язку, так і властивості процесу відновлення структурних кодових комбінацій. Проведемо порівняльну оцінка вірогідності інформації у випадку передачі вихідних даних у незжатому виді, передачі кодів одноознакових і двоознакових

чисел у двійковому структурному просторі.

Розроблений метод структурного відновлення дозволяє без перекручувань одержати вихідний фрагмент джерела повідомлення довільного алфавіту. Однак у випадку передачі кодових комбінацій стислих даних по каналах зв'язку з помилками, можуть виникнути перекручування, що впливають на вірогідність відновлюваної інформації.

Встановлено, що для достовірного одержання інформації, структурне компактне подання даних повинно мати не тільки властивість взаємооднозначного відновлення, але й бути завадостійким до помилок у каналі зв'язку. Розглянуті властивості завадостійкості двоознакових структурних кодів $N(m,\Lambda,\Theta^{(k)})_j$ у випадку їхньої передачі по каналу зв'язку з помилками.

Визначено, що в процесі двоознакового структурного кодування двійкових даних у структурному просторі для двійкової послідовності, елементи якої задовольняють системі обмежень формується код-номер $N(m,\Lambda,\Theta^{(k)})_j$:

$$A(m,\Theta^{(k)})_j=\{a_{11j},...,\ a_{m_1,1j},...,a_{1,\gamma,j},...,\ a_{m_\gamma,\gamma,j},...,a_{1Zj},...,\ a_{m_Z,Zj}\},\qquad(3.11)$$

$$0\le a_{ij}\le s_i,\ i=\overline{1,m};\ \ \vartheta=\sum_{z=1}^{Z}\vartheta_z^{(k)};\ \ \vartheta_z^{(k)}=\vartheta_{z,j},\ z=\overline{1,Z}.$$

При цьому в канал зв'язку, у випадку відсутності завадостійкого кодування, передається двійкове кодове подання $N(m,\Lambda,\Theta^{(k)})_2^{(j)}$ коду-номера $N(m,\Lambda,\Theta^{(k)})_j$:

$$N(m,\Lambda,\Theta^{(k)})_2^{(j)}=[\ell og_2 V(m,\Lambda,\Theta^{(k)})_j]+1.\qquad(3.12)$$

При передачі даних каналами зв'язку можуть виникнути помилки. Тоді буде прийняте кодове слово $A_L^{\bullet}$, що відрізняється від вихідного кодового слова A_L значеннями одного або більше розрядів $\nu(A_L)=n(\ell_\xi\ne\ell_\xi^{\bullet}),\ \xi=\overline{1,L}$, де $\nu(A_L)$ – кількість перекручених розрядів, рівних кількості розрядів $n(\ell_\xi\ne\ell_\xi^{\bullet})$, які відрізняють вихідні A_L й прийняте $A_L^{\bullet}$ кодові слова. У загальному випадку величина $\nu(A_L)$ змінюється в наступних межах: $0\le\nu(A_L)\le L$. Якщо $\nu(A_L)\ge 1$, то отримане на прийомній стороні значення коду-номера $N(m,\Lambda,\Theta^{(k)})_j^{\bullet}$ не буде

дорівнювати вихідному значенню коду-номера $N(m,\Lambda,\Theta^{(k)})_j$:

$$N(m,\Lambda,\Theta^{(k)})_j^{\bullet} \neq N(m,\Lambda,\Theta^{(k)})_j.$$

Встановлено, що величина спотворення значення коду-номера двоознакового структурного числа визначається значенням помилки $e(m,\Lambda,\Theta^{(k)})_j$, рівної абсолютному значенню різниці між величинами $N(m,\Lambda,\Theta^{(k)})_j^{\bullet}$ й $N(m,\Lambda,\Theta^{(k)})_j$:

$$e(m,\Lambda,\Theta^{(k)})_j = \left| N(m,\Lambda,\Theta^{(k)})_j^{\bullet} - N(m,\Lambda,\Theta^{(k)})_j \right|.$$

Згідно з системою правил двоознакового структурного кодування, формується код-номер $N(m,\Lambda,\Theta^{(k)})_j$ для двійкового структурного числа $A(m,\Theta^{(k)})_j$, отже, якщо $e(m,\Lambda,\Theta^{(k)})_j \geq 1$, то після проведення двоознакового структурного декодування відновлюється число $A(m,\Theta^{(k)})_j^{\bullet}$, що може відрізнятися від вихідної послідовності значеннями двійкових елементів.

Встановлено, що кількість ν двійкових елементів відновлених з помилкою дорівнює кількості двійкових елементів, для яких виконується нерівність $a_{ij} \neq a_{ij}^{\bullet}$, $i=\overline{1,m}$, Оскільки довжина двоознакового структурного числа дорівнює m, то кількість елементів ν, якими відрізняються вихідні $A(m,\Theta^{(k)})_j$ й відновлена $A(m,\Theta^{(k)})_j^{\bullet}$ двійкові послідовності, змінюється в межах $0 \leq \nu \leq m$. На основі аналізу виявлено, що помилки в кодовому слові A_L можуть відбутися як у не значимих ,так й у значимих розрядах. Не значимими є g старших розрядів коду A_L. Встановлено, що існує можливість виявити будь-яку кількість таких помилок у старших розрядах, для яких виконується умова

$$\ell_\xi = 1, \text{ для } N(m,\Lambda,\Theta^{(k)})_2^{(j)} + 1 \leq \xi \leq g + N(m,\Lambda,\Theta^{(k)})_2^{(j)} .$$

Показано, що при виникненні похибки, буде відновлена двійкова послідовність $A(m,\Theta^{(k)})_j^{\bullet}$, для якої $\nu \geq 1$ та знайдеться хоча б один двійковий елемент, яким будуть відрізнятися вихідні й прийнята двійкові послідовності. Тоді можливі два варіанти прояву помилок каналу зв'язку. Для першого варіанта цікавим є абсолютне значення, що представлено у двійковому виді послідовністю

$A(m,\Theta^{(k)})_j$. Для другого варіанта кожен компонент послідовності $A(m,\Theta^{(k)})_j$ розглядається як окремий елемент оброблюваних даних.

Встановлено межі відхилення коду-номера. Визначено, що для будь-якої кількості помилок, що виникли у двоознаковій структурної кодової конструкції (при її передачі по каналу зв'язку) величина $N_m^{\bullet}$ відхилення коду-номера перебуває в таких межах

$$\sum_{\xi=0}^{[\frac{m+1}{2}]} N(m,\vartheta_\xi)_{\min} \le N_m^{\bullet} \le 2^m - \sum_{\xi=0}^{[\frac{m+1}{2}]} N(m,\vartheta_\xi)_{\max}, \qquad (3.13)$$

де $N(m,\vartheta_\xi)_{\min}$, $N(m,\vartheta_\xi)_{\max}$ – значення коду-номера з урахуванням обмеження на число серій, що дорівнює ϑ_ξ, відповідно для початкової й кінцевої двійкової послідовності множини двоознакових структурних чисел у двійковому структурному просторі.

Визначено границі локалізації впливу помилки каналу зв'язку. Встановлено, що для будь-якої кількості помилок, що виникли у двоознаковій структурній кодовій конструкції (при її передачі по каналу зв'язку) для абсолютної величини ε відхилення від вихідного значення N_m, виконується нерівність

$$\varepsilon \le \sum_{\xi=0}^{[\frac{m+1}{2}]} N(m,\vartheta_\xi)_{\max} - \sum_{\xi=0}^{[\frac{m+1}{2}]} N(m,\vartheta_\xi)_{\min} = \sum_{\xi=0}^{[\frac{m+1}{2}]} \left(N(m,\vartheta_\xi)_{\max} - N(m,\vartheta_\xi)_{\min} \right) = \sum_{\xi=0}^{[\frac{m+1}{2}]} \sum_{i=1}^{m} (a_{ij}^{(\max)} - a_{ij}^{(\min)})\, p_{ij}^{(\xi)}. \qquad (3.14)$$

Встановлено, що властивість локалізації впливу помилки каналу зв'язку при декодуванні двоознакових структурних кодових комбінацій проявляється в обмеженні знизу й зверху можливих значень величини відхилення ε відновленого числа $N_m^{\bullet}$ від вихідного N_m, що задається нерівностями (3.13-3.14). Показником завадостійкості є значення кількості ν двійкових елементів відновлених з помилкою. Якщо довжина двоознакового числа дорівнює m, то величина ν змінюється в межах $0 \le \nu \le m$. Визначено, що для варіанта передачі по каналу зв'язку вихідних m- елементних послідовностей без стиснення ймовірність $P_{\nu,m}^{(0)}$ того (3.15), що помилка відбудеться у ν двійкових елементах

дорівнює

$$P_{\nu,m}^{(0)} = n_{m,\nu}^{(0)}\, p_0^{\nu}\,(1-p_0)^{m-\nu} \quad , \tag{3.15}$$

де

$n_{m,\nu}^{(0)}$ – кількість розміщень ν перекручених двійкових елементів у послідовності довжиною m: $n_{m,\nu}^{(0)} = m!/(\nu!\,(m-\nu)!)$;

q_0 – імовірність події, що полягає у тому, що у двійковому елементі відбудеться помилка $q_0 = (1-p_0)$.

У випадку передачі двійкової послідовності у вигляді одноознакового структурного числа $A(m,\vartheta)_j$ по каналу зв'язку передається кодове подання номера $N(m,\vartheta)_j$ (рисунок 3.10).

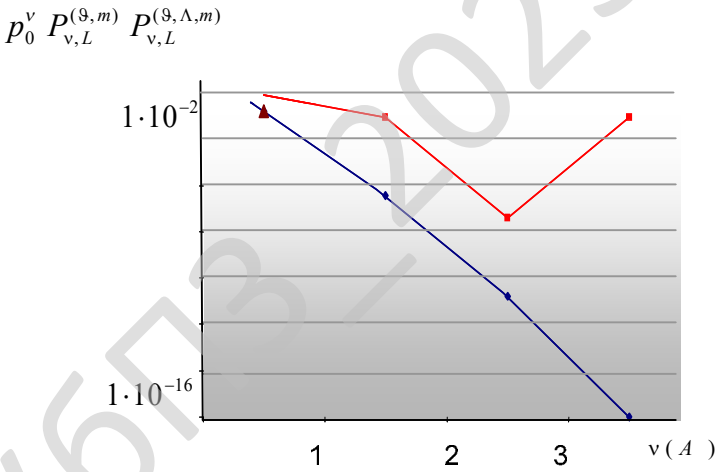

Рисунок 3.10 – Графіки залежності p_0^{ν}, $P_{\nu,L}^{(\vartheta,m)}$, $P_{\nu,L}^{(\vartheta,\Lambda,m)}$ (у логарифмічному масштабі) від $\nu(A_L)$ для $p_0 = 10^{-2}$ при відновленні: 1 – вихідних послідовностей без стиснення, $m=8$; 2 – одноознакових структурних чисел для $\vartheta=4$, і $m=8$; 3 – двоознакового структурного числа для $\vartheta=4$, $\lambda_2=1$, $\lambda_4=1$, $m=8$

Визначено, що серед помилок різної кратності найбільшу ймовірність мають однократні помилки. Це відповідає, як вихідним послідовностям, які передаються по каналу зв'язку не в стислому виді, так й одноознаковим і двоознаковим структурним числам.

Порівняльні оцінки ймовірностей p_0^{ν}, $P_{\nu,L}^{(\vartheta,m)}$, $P_{\nu,L}^{(\vartheta,\Lambda,m)}$ появи помилок кратності ν для випадку передачі каналами зв'язку вихідних двійкових даних у не стисненому виді, передачі кодових конструкцій одноознакових і двоознакових структурних чисел у двійковому просторі наведені на рисунку 3.11.

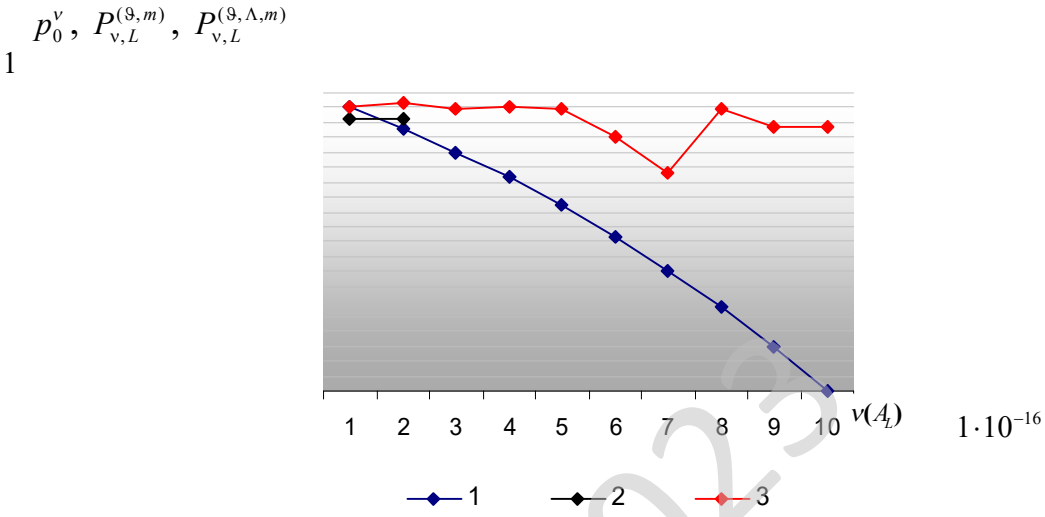

Рисунок 3.11 – Графіки залежності p_0^{ν}, $P_{\nu,L}^{(\vartheta,m)}$, $P_{\nu,L}^{(\vartheta,\Lambda,m)}$ (у логарифмічному масштабі) від $\nu(A_L)$ для $p_0=10^{-2}$ при відновленні: 1 – вихідних послідовностей без стиснення, $m=10$; 2 – одноознакових структурних чисел для $\vartheta=5$, і $m=10$;3 – двоознакових структурних чисел для $\vartheta=5$, $\lambda_3=1$, $m=10$

На основі аналізу наведених залежностей визначено, що ймовірність однократних помилок при відновленні одноознакових структурних чисел перевищує ймовірність однократних помилок у вихідних нестиснених послідовностях у середньому від 1,3 до 2,27 разів.

Встановлено, що із збільшенням довжини структурного числа програш по ймовірності однократних помилок для одноознакових структурних чисел скорочується в середньому в 2 рази. Визначено, що при відновленні двоознакових кодових конструкцій імовірності помилок великого ступеня кратності (від трьох і вище) дорівнюють нулю.

Мінімальний ступінь кратності помилок, починаючи з якої ймовірності їхньої появи будуть дорівнювати нулю, залежить від довжини оброблюваної послідовності й від структурних обмежень, що накладають на їхні елементи. Ця закономірність викликана тим, що при накладенні обмежень на позиції із припустимою появою одиниць відбувається скорочення кількості допустимих послідовностей, які відрізняються між собою різною кількістю двійкових елементів; зменшення максимального ступеня кратності помилок, якими можуть відрізнятися між собою двоознакові структурні числа.

Отже, на відміну від одноознакових чисел, двоознакові структурні числа мають властивість самокорекції помилок.

У цьому випадку ймовірність і кількість помилок після відновлення двоознакових структурних чисел буде меншою, ніж ймовірність і кількість помилок у кодовому слові отриманому каналами зв'язку.

Тому для ймовірності помилки одного розряду, не перевищуючої 10^{-4}, кодові конструкції двоознакових структурних чисел допускається передавати по каналу зв'язку без внесення додаткових коригувальних розрядів (без зниження ступеня стиснення). За рахунок властивостей самокорекції помилок великого ступеня кратності, допускається використання меншої кількості коригувальних розрядів або повна їх відсутність.

Встановлено, що залежно від структурних характеристик двоознакові кодові конструкції можуть мати більшу завадостійкість у порівнянні з випадком передачі каналами зв'язку нестиснених двійкових послідовностей. Це проявляється в зменшенні ймовірності однократних помилок і рівності нулю помилок великої кратності.

У шостому розділі розроблено технологію підвищення ефективності функціонування АСУ на основі методів двоознакового структурного кодування даних, методів оцінок й корекції завадостійкості двоознакових кодових конструкцій, що дозволило забезпечити необхідний час обробки й неспотворену передачу визначених обсягів даних (бітових потоків) сформованих від різних

джерел інформації. Розроблено програмно-апаратний комплекс підсистеми стиснення інформаційних потоків у складі структурно-логічної схеми функціонування АСУ, залежно від характеристик процесів обробки й подання даних.

Проведено класифікацію цієї системи, установлено її мета й завдання щодо підвищення завадостійкості та вірогідності інформації, швидкодії процесів обробки і передачі (розподілу) даних без втрат для забезпечення процесу якісного прийняття рішень в АСУ.

Обґрунтовано роль і місце розробленої підсистеми стиснення інформаційних потоків на основі двоознакового структурного кодування й декодування даних.

На основі розробленої технології показано шляхи побудови та підвищення ефективності якості функціонування АСУ з урахуванням кількісних оцінок параметрів підсистеми стиснення даних, обраних показників якості і сформульованих критеріїв у вигляді функціональної схеми системи (рисунок 3.12). На базі визначеної технології розроблена структурно-логічна схема АСУ, що містить у собі велику кількість розташованих на значній відстані елементів і підсистем, у тому числі й територіально рознесені космічні сегменти.

Запропоновану АСУ за видом об'єкта діяльності можливо класифікувати, як багатофункціональну інтегровану систему управління виробничо-господарською діяльністю, де обробляються дані як про технологічні процеси, так і про виробничо-господарську діяльність.

Визначено, що ґрунтуючись на розроблених методах і програмно-апаратному комплексі й розв'язуваних ними завдань щодо забезпечення підвищення коефіцієнта стиснення й додаткового скорочення часу на обробку даних, підсистему стиснення інформаційних потоків можна класифікувати як систему, що забезпечує рішення завдань функціональних підсистем АСУ.

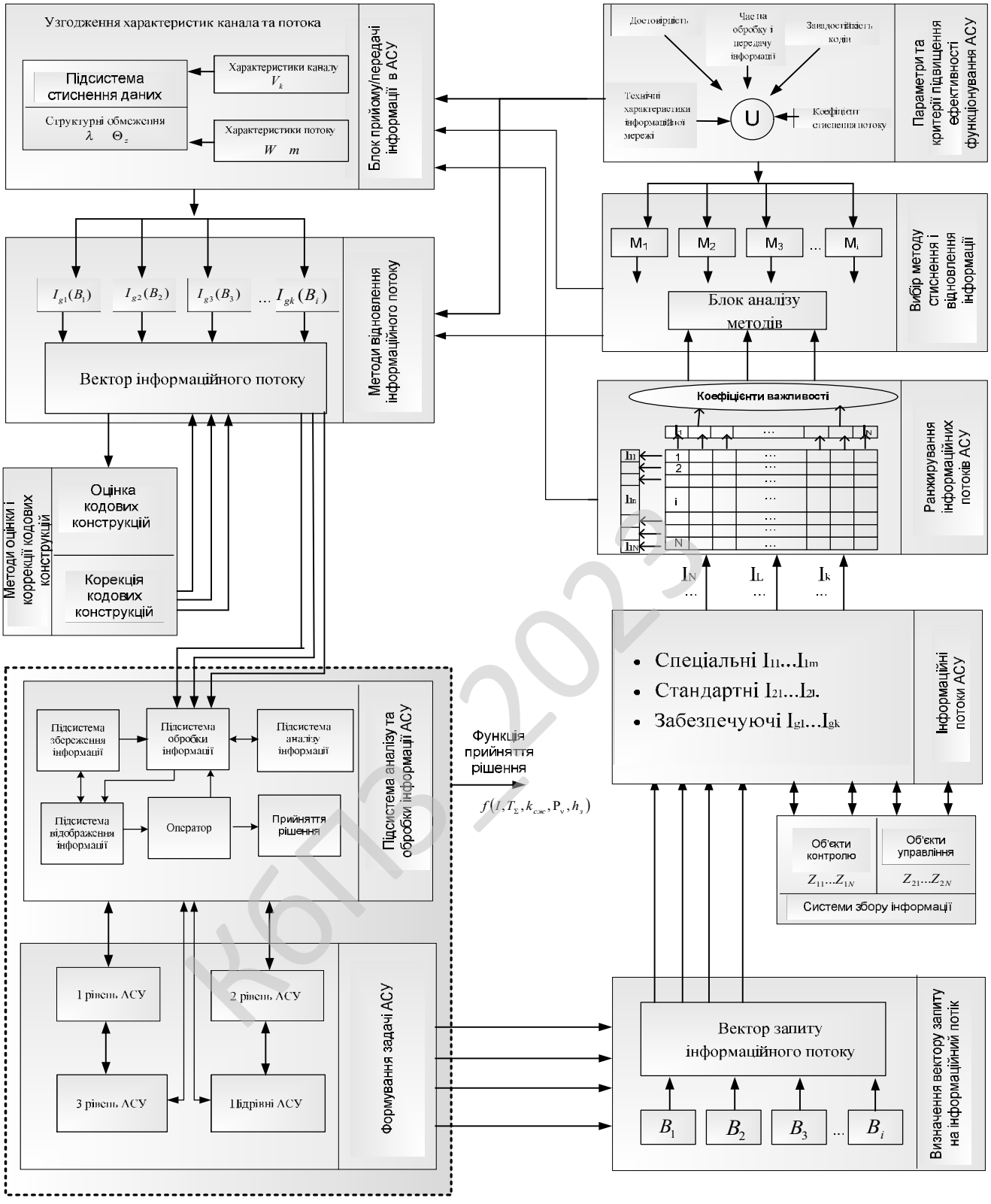

Рисунок 3.12 – Функціональна схема системи

Показано, що для розробленого методу компактного подання даних на основі двоознакового структурного кодування сумарний час на обробку й передачу реалістичних зображень (обсягом $W^{(\inf)}=172{,}8$ Мбіт, швидкість обробки $U_{обр}=10^{10}$ оп/с, швидкість передачі даних по інформаційним мережам – 2,048 Мбіт/с $\le U_n \le$ 61 Мбіт/с) змінюється в межах від 0,04 до 20 с.

При цьому виграш за часом обробки й передачі даних для розробленого методу щодо відомих досягає в середньому від 2.1 до 4.2 разів залежно від співвідношення характеристик комп'ютерних мереж АСУ.

Визначено, що величина сумарного часу процесу $T(W^{(\inf)})_{\Sigma}$ функціонально залежна від коефіцієнту стиснення даних $k_{ст.}$. Величина $k_{ст.}$ визначається, як відношення обсягу $W^{(\inf)}$ вихідних даних до обсягу $W^{(c)}$ стиснених даних.

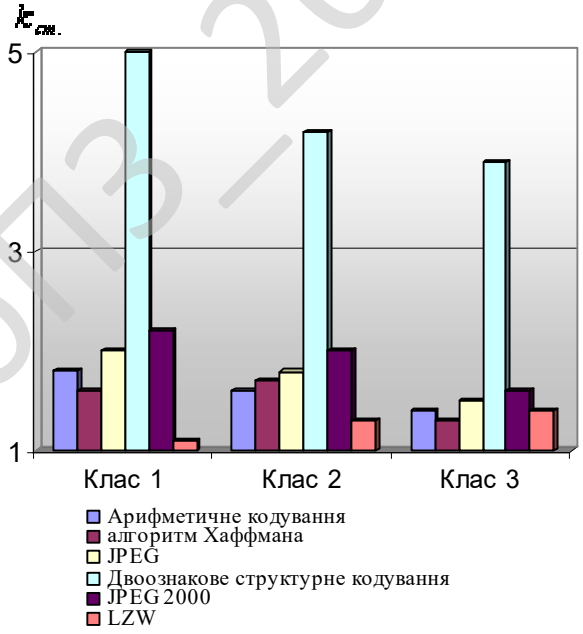

Рисунок 3.13 – Діаграми залежності коефіцієнта стиснення $k_{ст.}$ при обробці різних видів зображень: 1 класу – реалістичні, 2 класу- реалістичні, 3 клас – штучні

Отже: ступінь стиснення для двоознакового структурного кодування двійкових даних у структурному просторі, залежно від виду оброблюваних даних перебуває в межах від 2,2 до 5 разів.

При цьому найбільше скорочення обсягів даних досягається при обробці реалістичних зображень попередньо представлених у спектральному виді (до 6.25 разів).

При цьому, для розробленого методу компактного подання в порівнянні з відомими методами, виграш по величині стиснення в середньому дорівнює 2,5 рази.

Показано, що важливим є показник вірогідності одержуваних даних. У роботі обґрунтовано, як показник якості процесів в АСУ, ймовірність перекручування P_ν заданої кількості ν двійкових елементів у кодових комбінаціях.

На основі введення цього показника розроблені методи оцінок й корекції завадостійкості двійкових послідовностей, які дають можливість локалізувати вплив і уможливити процедуру самокорекції помилок в кодових словах при їх передачі в комп'ютерних мережах АСУ.

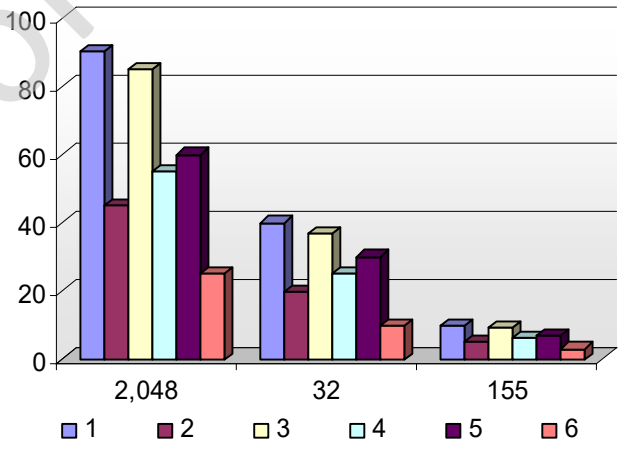

Рисунок 3.14 – Діаграми залежності часу $T(W^{(\inf)})_\Sigma$ при обробці та передачі реалістичних зображень (швидкість U_n Мбіт/с) для: 1 – без стиснення; 2 – методи формату JPEG; 3 – методи формату BMP; 4 –архіватора rar; 5 – архіватора zip; 6 – розробленого методу компактного подання

Встановлено, що базуючись на проведених дослідженнях та відповідно до висунутих вимог і особливостей рішення завдань стиснення даних в комп'ютерних мережах АСУ було досягнуто поставлених умов для визначених критеріїв ефективності функціонування АСУ:

1) у випадку відомого часу t, необхідного для передачі обсягу інформації $W^{(\inf)}$ – відповідність між необхідною величиною t й часом $T(W^{(\inf)})_{\Sigma}$ виду:

$$T(W^{(\inf)})_{\Sigma} \to t. \qquad (3.16)$$

забезпечується технологією компактного подання даних на основі двоознакового структурного кодування й декодування даних.

2) у загальному випадку – досягається мінімізація часу обробки й передачі даних

$$T(W^{(\inf)})_{\Sigma} \to \min. \qquad (3.17)$$

З виконанням співвідношень (3.16) і (3.17) виконуються і умови обмежень, що накладаються на швидкість обробки і передачі даних по комп'ютерним мережам АСУ, а також виконуються вимоги по забезпеченню ступеню вірогідності інформації.

$$T(W^{(\inf)})_{\Sigma} = \frac{\nu(W^{(\inf)})_k}{U_{обр}} + \frac{W^{(\inf)}}{U_n\ k_{ст}} + \frac{\nu(W^{(\inf)})_d}{U_{обр}} \ \to \ min, \qquad (3.18)$$

$$\begin{cases} h \ge h_з;\ U_n \le U_{зад,n}; \\ U_{обр} \le U_{зад,обр};\ W_{взу} \le W_{зад}, \end{cases} \qquad (3.19)$$

де

$h_з$ – задане значення відношення сигнал/шум;

$U_{зад,обр}, U_{зад,n}, W_{зад,}$ – відповідно задані для конкретної комп'ютерної мережі – швидкість виконання машинних операцій, швидкість передачі даних й заданий обсяг даних.

Показано, що в результаті компактного подання даних досягається зниження їх обсягів, що приводить до скорочення кількості пакетів даних, зниження ступеня інтенсивності завантаження каналів зв'язку, зменшення часу

затримки пакетів на транзитивних вузлах комп'ютерних мереж АСУ.

Реалізація програмно-апаратного комплексу підсистеми стиснення даних на базі двоознакового структурного кодування, забезпечує підвищення ефективності функціонування АСУ на основі відповідності заданого часу обміну оперативною інформацією, її вірогідності.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.15. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

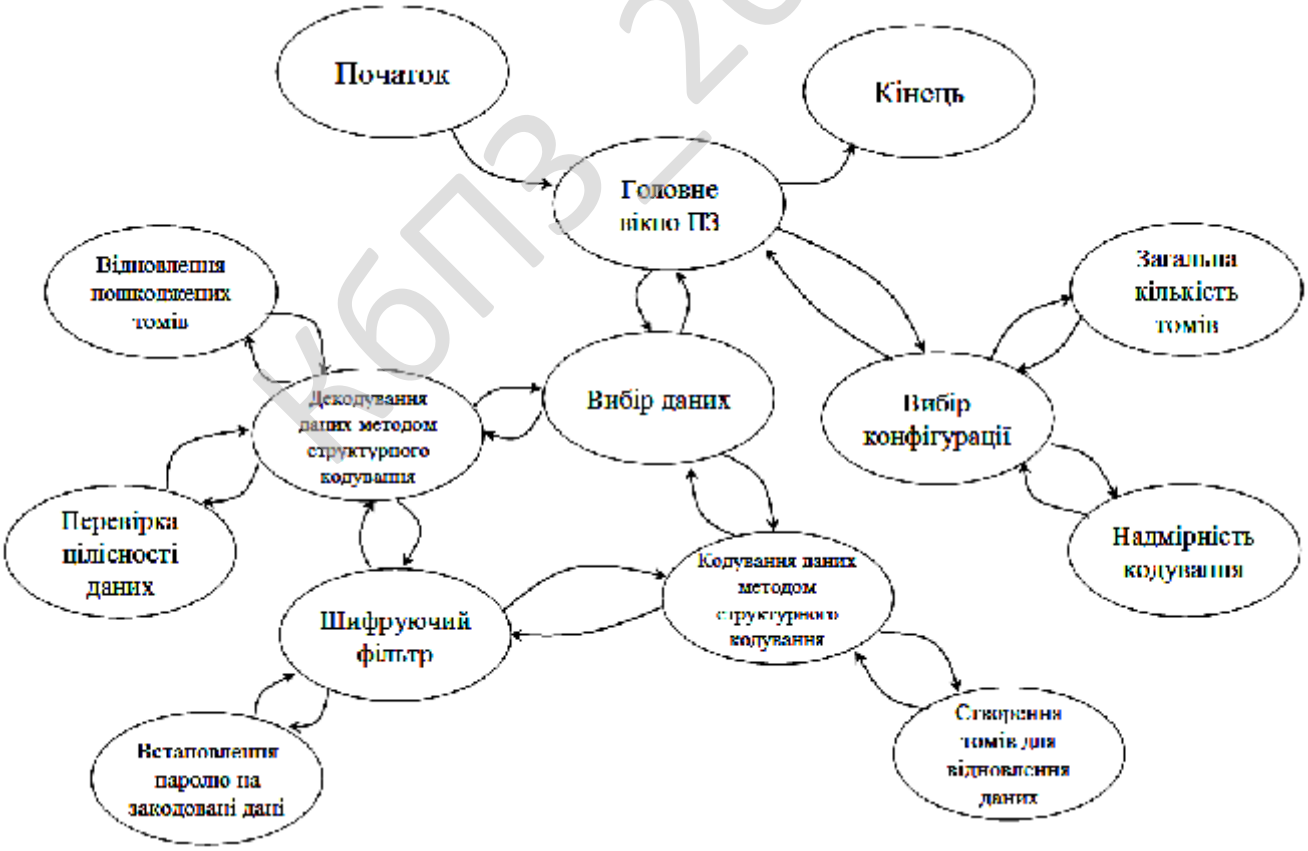

Рисунок 3.15 – Діаграма взаємодії процесів

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

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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 ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 та 4.3 зображено роботу підпрограм.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограм та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограм виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю методів структурного кодування даних у комп’ютерних мережах автоматизованих систем управління.

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

функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

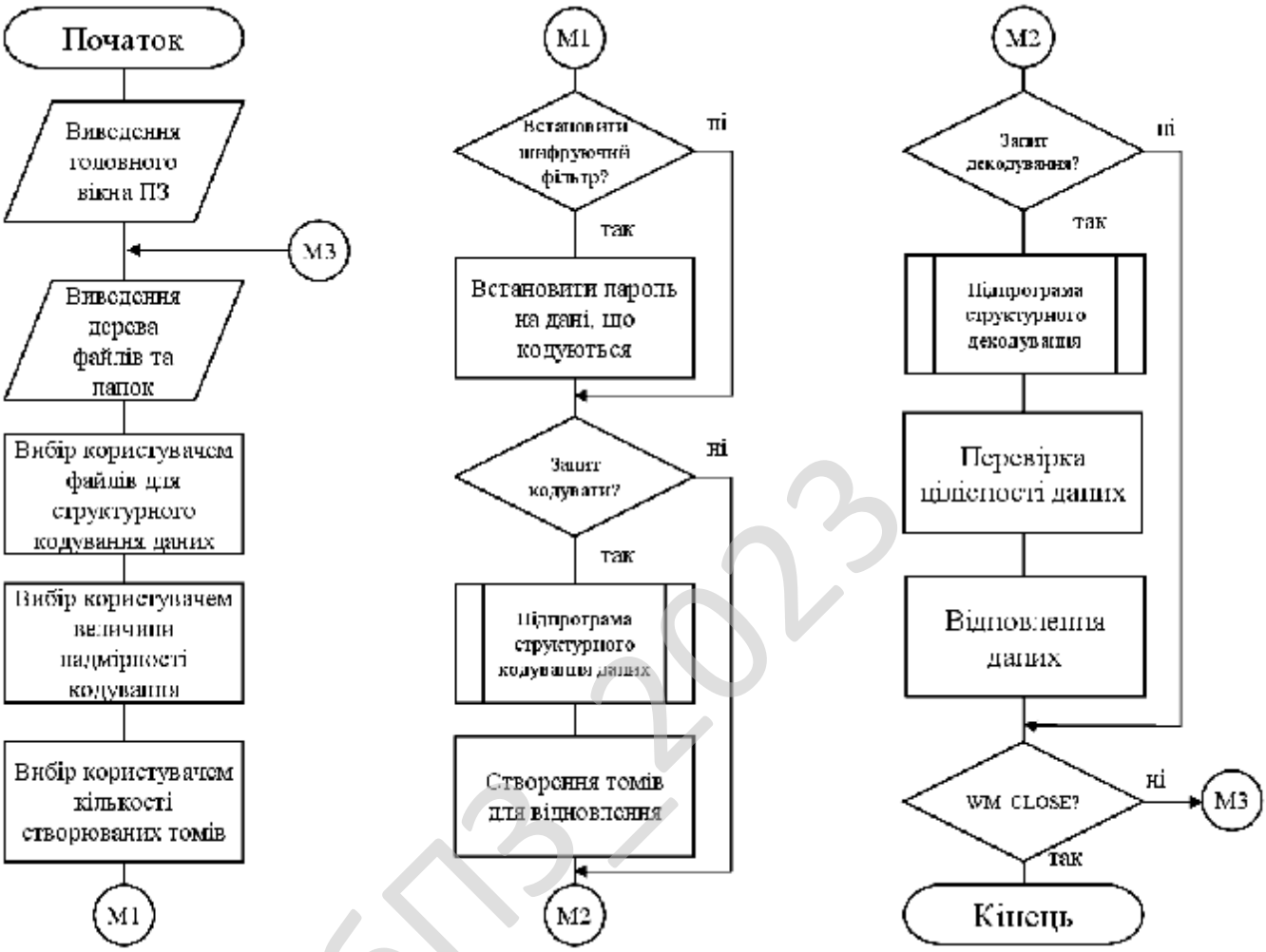

Рисунок 4.1 – Блок-схема основної програми

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем.

UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

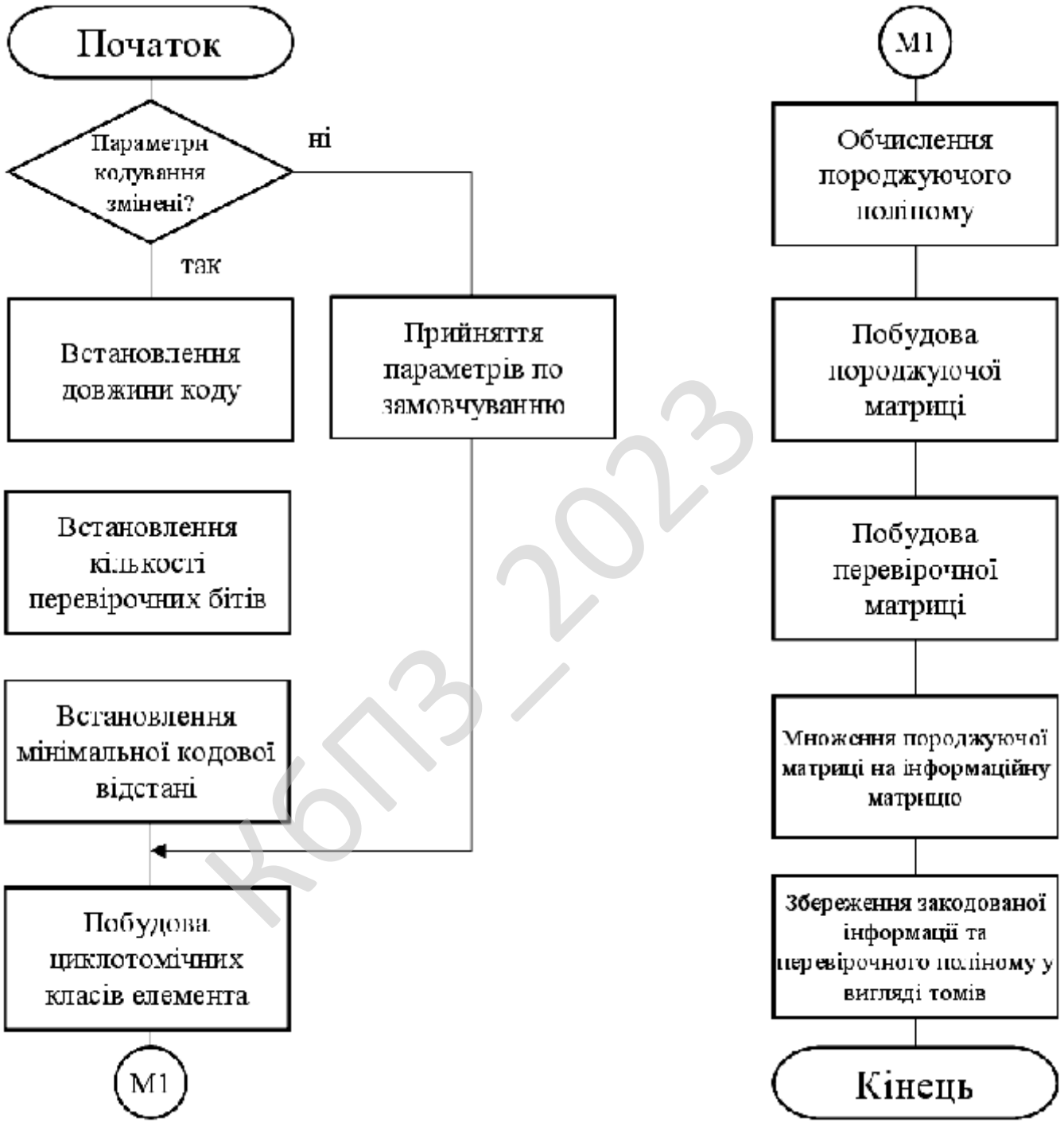

Рисунок 4.2 – Блок-схема роботи підпрограми

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

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

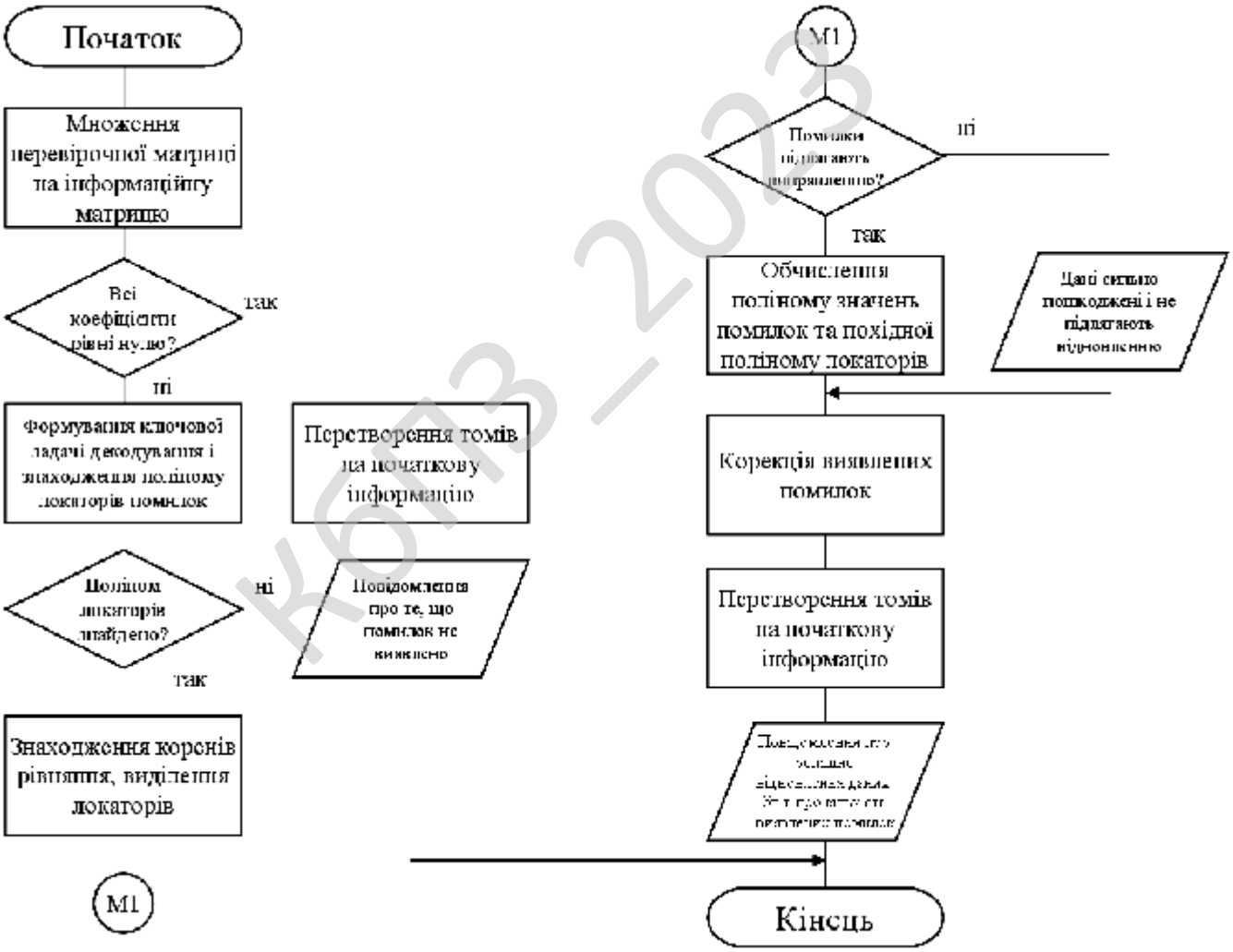

Рисунок 4.3 – Блок-схема роботи підпрограми

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

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

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

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

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

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

– Програмістам які реалізовують модулі інформаційної системи.

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

Нижче наведемо ту частину коду, яка виконує вищеперераховані дії.

```
namespace RD
{
/// <summary>
/// Клас кодера циклічного коду
/// </summary>
```

```
    public class CicleCodeEncoder : CicleCodeBase
    {
        #region Construction & Destruction
/// <summary>
/// Конструктор кодера за замовчуванням
/// </summary>
        public CicleCodeEncoder()
        {
// 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 CicleCodeEncoder(int dataCount, int eccCount)
        {
// Установка конфігурації кодера
            SetConfig(dataCount, eccCount, (int)CicleCodeType.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 циклічного коду (по типу матриці)</param>
        public CicleCodeEncoder(int dataCount, int eccCount, int codecType)
        {
// Установка конфігурації кодера
SetConfig(dataCount, eccCount, codecType);
// Створюємо об'єкт класу роботи з елементами поля Галуа
            this.eGF16 = new GF16();
        }
        #endregion Construction & Destruction
        #region Public Operations
        /// <summary>
/// Установка конфігурації кодера
        /// </summary>
/// <param name="dataCount">Кількість основних томів</param>
```

```
/// <param name="eccCount">Кількість томів для відновлення</param>
/// <param name="codecType">Тип кодека кодера циклічного коду (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)CicleCodeType.Dispersal)
            {
      maxVolCount = (int)CicleCodeConst.MaxVolCountDisp;
       } else
     {
     maxVolCount = (int)CicleCodeConst.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.eCicleCodeType)
        )
{
        this.mainConfigChanged = true;
}
// Зберігаємо конфігурацію
    this.n = dataCount;
    this.m = eccCount;
    this.eCicleCodeType = 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.eCicleCodeType == (int)CicleCodeType.Alternative)
                {
                    this.iterOfFirstStage = m;
                } else
                {
                    this.iterOfFirstStage = ((n * m) * n) + (n * ((n + m) + (n *
(n + m))));
                }
                this.iterOfSecondStage = 0;
// У кодери немає інвертування матриці
                this.configIsOK = true;
            } else
            {
//...указуємо на помилку конфігурації
                this.configIsOK = false;
            }
            return this.configIsOK;
        }
        /// <summary>
/// Метод множення матриці кодування на вхідний прологарифмований вектор
        /// </summary>
        /// <param name="dataLog">Прологарифмований вхідний вектор (вихідні
дані)</param>
        /// <param name="ecc">Вихідний вектор (надлишкові дані)</param>
        /// <returns>Булевський прапор результату операції</returns>
        public bool Process(int[] dataLog, ref int[] ecc)
        {
// Якщо кодер зконфігуровано некоректно, обробка неможлива!
            if (!this.configIsOK)
            {
                return false;
            }
// Копіюємо покажчик на масив експонент для скорочення часу обігу
            int[] GF16Exp = this.eGF16.GFExpTable;
            // Обчислення результату множення матриці на вектор
            for (int i = 0; i < this.m; i++)
            {
                int mulSum = 0;
```

```
// Сума добутку рядка матриці на стовпець
                int i_n = i * this.n;
// Зсув у масиві до елементів i-ой рядка
                for (int j = 0; j < this.n; j++)
                {
                    mulSum ^= GF16Exp[this.FLog[i_n + j] + dataLog[j]];
                }
                ecc[i] = mulSum;
            }
            return true;
        }
        #endregion Public Operations
        #region Private Operations
        /// <summary>
        /// Заповнення матриці Вандермонда даними
        /// </summary>
        protected override void FillFLog()
        {
            // Якщо основна конфігурація змінилася...
            if (this.mainConfigChanged)
            {
                if (this.eCicleCodeType == (int)CicleCodeType.Dispersal)
                {
//...робимо формування дисперсної матриці "D"
                    if (!MakeDispersalMatrix())
                    {
// Указуємо, що кодер зконфігуровано некоректно
                        this.configIsOK = false;
// Активуємо індикатор актуального стану змінних-членів
                        this.finished = true;
// Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();
                        return;
                    }
                } else
                {
//...робимо формування альтернативного заповнення матриці "A"
                    if (!MakeAlternativeMatrix())
                    {
// Указуємо, що кодер зконфігуровано некоректно
                        this.configIsOK = false;
// Активуємо індикатор актуального стану змінних-членів
```

```
                        this.finished = true;
// Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();
                        return;
                    }
                }
// Виділяємо пам'ять під матрицю "FLog"
                this.FLog = new int[this.m * this.n];
// Заповнюємо матрицю кодування
                for (int i = 0; i < this.m; i++)
                {
// Зсув у масиві до елементів i-ой рядка
                    int i_n = i * this.n;
                    // Залежно від типу кодера беремо дані з відповідного масиву
                    if (this.eCicleCodeType == (int)CicleCodeType.Dispersal)
                    {
// Формування рядка в матриці кодування
                        for (int j = 0; j < this.n; j++)
                        {
// У матрицю кодування поміщаємо логарифми її вихідних елементів
// (для прискорення множення матриці на вектор)
                            this.FLog[i_n + j] = this.eGF16.Log(this.D[((this.n +
i) * this.n) + j]);
                        }
                    } else
                    {
// Формування рядка в матриці кодування
                        for (int j = 0; j < this.n; j++)
                        {
                            int idx = i_n + j;
// У матрицю кодування поміщаємо логарифми її вихідних елементів
// (для прискорення множення матриці на вектор)
                            this.FLog[idx] = this.eGF16.Log(this.A[idx]);
                        }
                    }
// У випадку, якщо потрібна постановка на паузу, подію "executeEvent"
// буде скинуто, і будемо на паузі аж до його появи
                    ManualResetEvent.WaitAll(this.executeEvent);
                    // Якщо зазначено, що потрібно вийти з потоку – виходимо
                    if (ManualResetEvent.WaitAll(this.exitEvent, 0, false))
                    {
// Указуємо, що кодер зконфігуровано некоректно
```

```
                        this.configIsOK = false;
// Активуємо індикатор актуального стану змінних-членів
                        this.finished = true;
// Установлюємо подію завершення обробки
                        this.finishedEvent[0].Set();
                        return;
}
}
// Якщо є передплата на делегата завершення...
                if (OnCicleCodeMatrixFormingFinish != null)
                {
//...повідомляємо, що екземпляр класу готовий до роботи
                    OnCicleCodeMatrixFormingFinish();
                }

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

// Якщо є передплата на делегата завершення...
            if (OnCicleCodeMatrixFormingFinish != null)
            {
//...повідомляємо, що екземпляр класу готовий до роботи
                OnCicleCodeMatrixFormingFinish();
            }
// Активуємо індикатор актуального стану змінних-членів
            this.finished = true;
// Установлюємо подію завершення обробки
            this.finishedEvent[0].Set();
        }
        #endregion Private Operations
    }
}
```


4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використати фіналіста конкурсу AES – шифр Rijndael. Він є нетрадиційним блоковим шифром, оскільки не використовує мережу Фейштеля для

криптоперетворень. Алгоритм представляє кожний блок кодуємих даних у вигляді двовимірного масиву байт розміром 4x4, 4x6 або 4x8 залежно від установленої довжини блоку. Далі на відповідних етапах перетворення відбуваються або над незалежними стовпцями, або над незалежними рядками, або взагалі над окремими байтами в таблиці.

Всі перетворення в шифрі мають строге математичне обґрунтування. Сама структура й послідовність операцій дозволяють виконувати даний алгоритм ефективно як на 16-бітних так і на 64-бітних процесорах. У структурі алгоритму закладена можливість паралельного виконання деяких операцій, що на багатопроцесорних робочих станціях може ще підняти швидкість шифрування в 4 рази.

Алгоритм складається з деякої кількості раундів (від 10 до 14 – це залежить від розміру блоку й довжини ключа), у яких послідовно виконуються наступні операції :

ByteSub – Таблична підстановка 8x8 біт (рисунок 4.4).

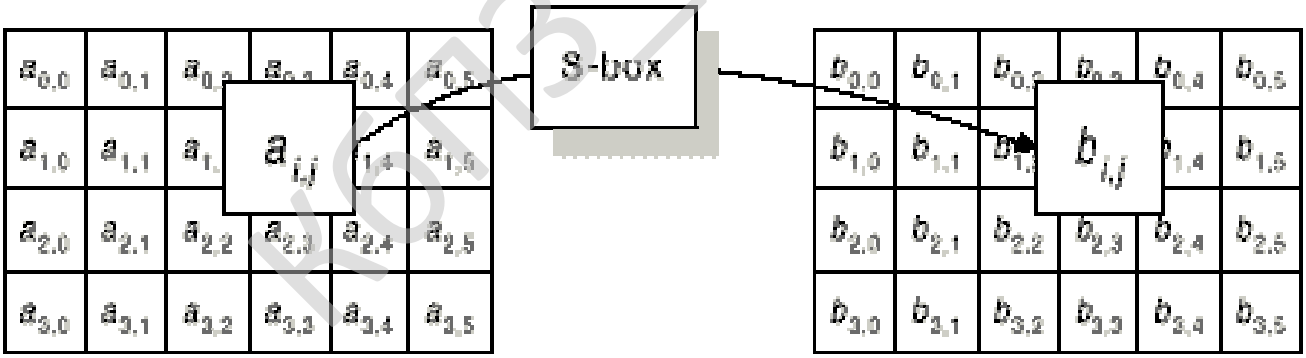

Рисунок 4.4 – Таблична підстановка 8x8 біт

ShiftRow – зрушення рядків у двовимірному масиві на різні зсуви (рисунок 4.5).

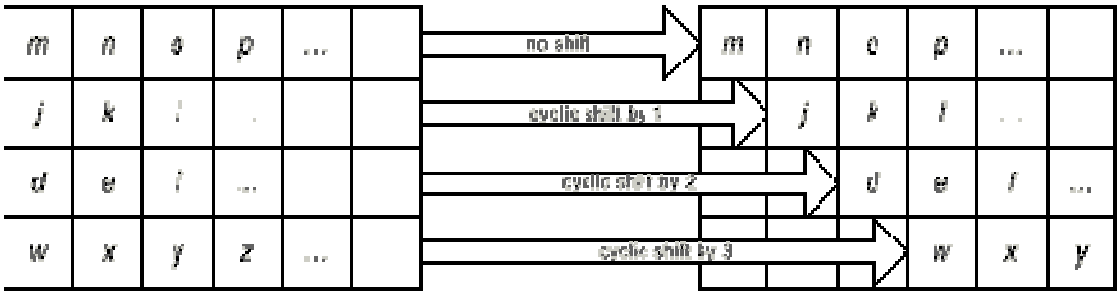

Рисунок 4.5 – Зрушення рядків у двовимірному масиві на різні зсуви

MixColumn – математичне перетворення, що перемішує дані усередині стовпця (рисунок 4.6).

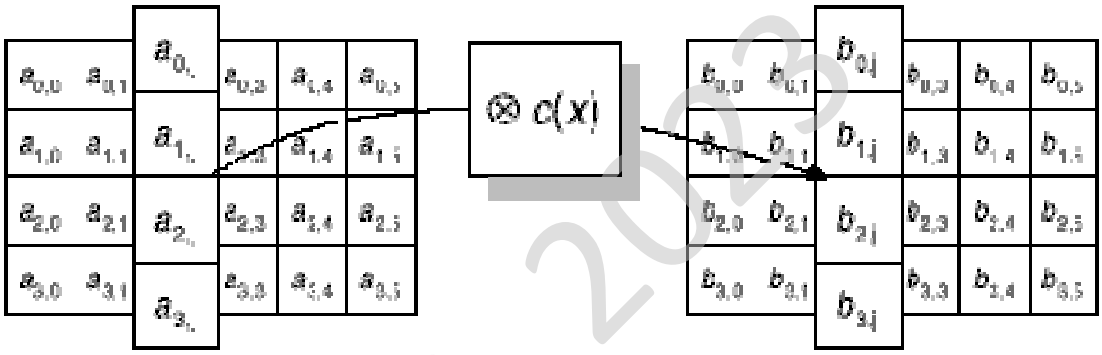

Рисунок 4.6 – Математичне перетворення, що перемішує дані усередині стовпця

AddRoundKey – додавання матеріалу ключа операцією XOR (рисунок 4.7).

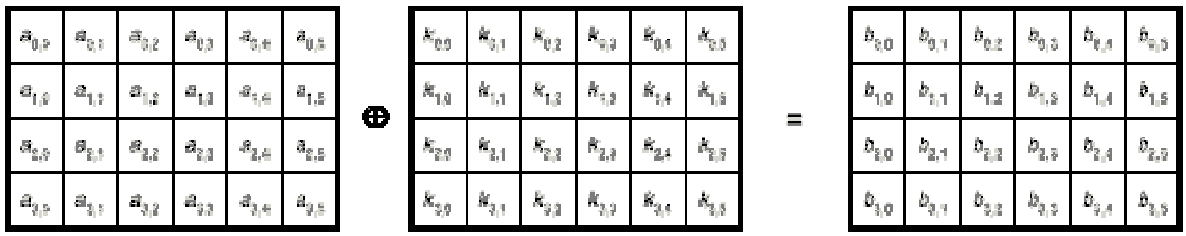

Рисунок 4.7 – Додавання матеріалу ключа операцією XOR

В останньому раунді операція перемішування стовпців відсутня, що робить всю послідовність операцій симетричною.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання магістерської роботи.

Розроблене програмне забезпечення методів структурного кодування даних у комп'ютерних мережах автоматизованих систем управління складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Дані; Додати; Налаштування; Довідка.

– Блок функціональних кнопок ПЗ.

– Вікно виведення результату роботи системи.

– Блок кнопок дії ПЗ.

– Навігаційного меню яке визивається натисканням правої клавіші маніпулятора миші.

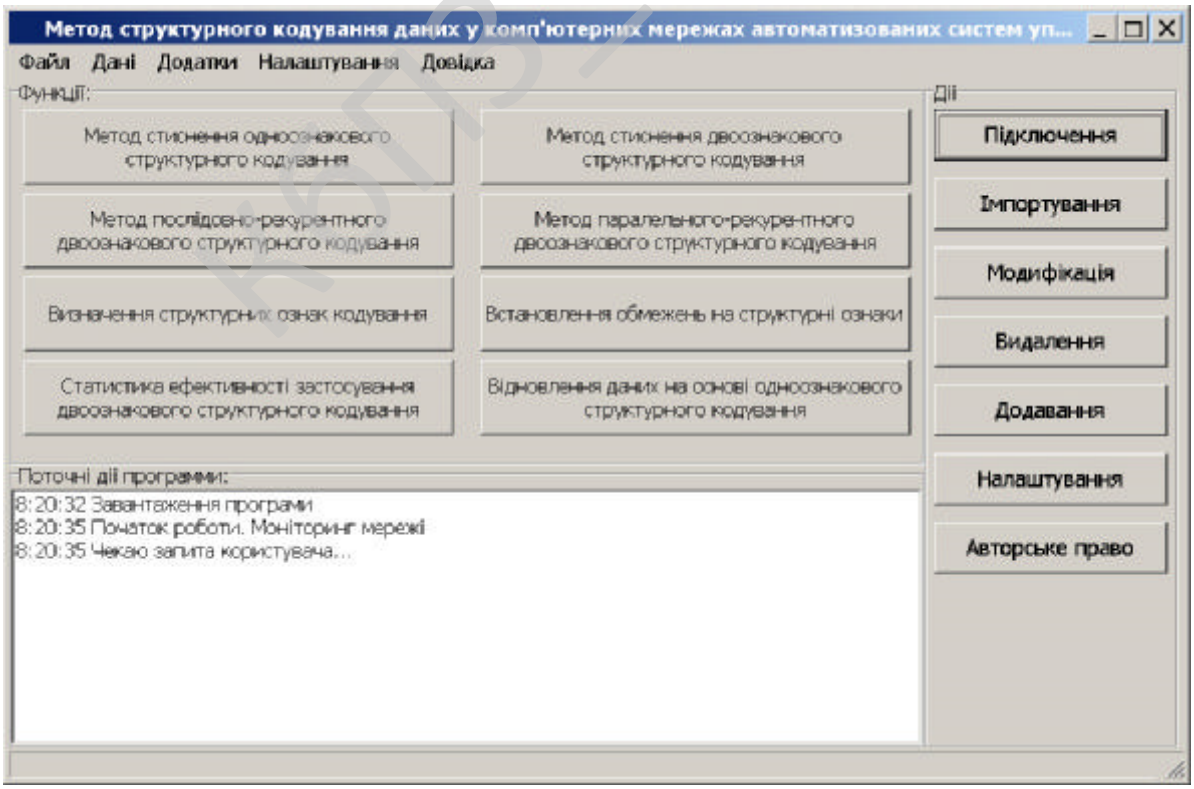

Рисунок 5.1 – Головне вікно розробленого ПЗ

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

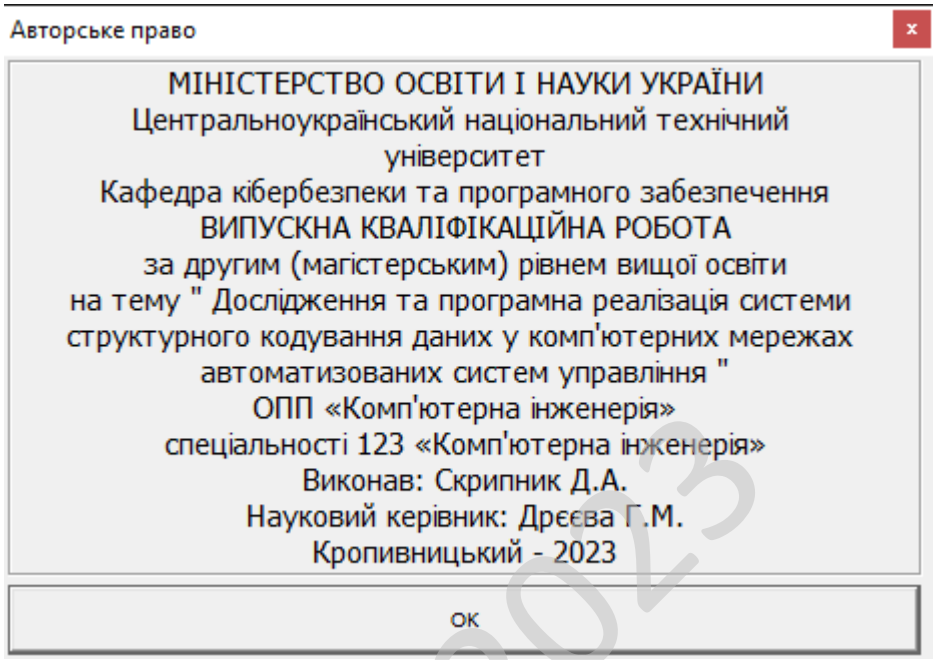

Рисунок 5.2 – Вікно розробника ПЗ

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

Проводилось тестування форматом чорної скриньки

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

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

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

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

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

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

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

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

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

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок. Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок: Некоректних чи відсутніх функцій; Помилок інтерфейсу; Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних; Помилок характеристик (необхідна ємність пам'яті і т.д.); Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів. Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення. Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються. Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Метою розробки є дослідження та програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Об'єктом дослідження є процес структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Предметом дослідження є методи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

Методи дослідження базуються на методах теорії кодування, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

– Розроблено вітчизняний продукт структурного кодування даних у комп'ютерних мережах автоматизованих систем управління, який має більш широкі можливості, на відміну від існуючих аналогів.

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 проведено дослідження та виконана програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління.

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	150
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)	–	1
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.	–	15000
33. Норматив додаткової зарплати, % :	$Н_д$	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	60
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.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

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

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}02\ 7.$$

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

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

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

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

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

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:

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

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)%.

$$T_{РП} = 0{,}3 \cdot 3{,}23 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 86 = 168 \text{ люд/день.}$$

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

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	168	Ф 7.1- 7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

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

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

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3,8 \text{ ставки.}$$

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 облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м.п.	2,5	100	250	4
Кабельне господарство електромереж	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 3}{1,2} = 567,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 567,5/(60 \cdot 8) = 1,2 \text{ ставки.}$$

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 (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,5	0,25
	Налаштування і конфігурування базової станції безпровідного 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	0,5	
Всього		2	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	0,5	0,2
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	

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	15000	45000
Продакт-менеджер	0,25	15000	11250
Інженер-програміст	3,8	15000	171000
Інженер-електронщик	1,2	15000	54000
Адміністратор мережі	0,25	15000	11250
Дизайнер WEB	0,2	15000	9000
Інженер-верстальник	0,2	15000	9000
Бухгалтер-економіст	0,136	15000	6120
Всього за період розробки	$R_{сп}$ = 7,036	-	$Ф_{роб}$ = 316620

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

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

$$З_{сд} = \frac{316620}{7 \cdot 60} = 754 \; грн.$$

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:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

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

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

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:

$$I_{нв} = R^1_{сп} \cdot Ц_м, \qquad (7.10)$$

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.6.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет-магазину Компбест за 29.10.23. (джерело https://compbest.com.ua.)

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10747
Системний блок	HP ProDesk 600 G2 Tower	7347
Процесор	6th gen Intel® Core™ i3 i3-6100 (3M Cache, 2 (4) ядра по 3.7 GHz)	-
Системна плата	Intel® Q150, PCI Express x16 – 1, PCI Express x1 – 3, DDR4-SDRAM, 4x USB 2.0, 6x USB 3.0, 4x Audio Ports, RJ-45 (LAN), VGA, 2x DP, COM-Port, 2x PS/2.	-
Відеокарта	Вбудована Intel® HD Graphics 530	-
Жорсткий диск	HDD: 500 Gb 7200 Serial ATA	-
Оперативна пам’ять	8GB (2133 MHz) DDR4-SDRAM (2 x 4 GB)	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	-
Корпус	ATX Middle Tower Pro, 3GTLA-489, PSU 450W(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	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	-
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1, 170/160, D-SUB, Wide)	3400
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

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

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	8	10747	8597,6	94573,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	113125,1

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	113125	-	-
Всього по групі	113125	40	45250
Нематеріальні активи			
4. Нематеріальні активи	15000	10	1500
Група 5, 6			
5. Вимірювальні пристрої	5190	25	1297,5
6. Транспортні засоби	0	20	0,0
7. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
Разом	$К_р$ = 1569315		$А_р$ = 125447,5

Примітка: вартість транспортного засобу приймаємо рівним нулю.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 754 \cdot 209/150 = 1050 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 1050 \cdot 10 \cdot 0{,}01 = 105 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(1050+105) = 254 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 1050 \cdot 15 \cdot 0{,}01 = 158 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картриджів, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно виданих норм приймаємо 0,6 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 200$ грн., визначаємо вартість паперу за період розробки $N_м = 3$ міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 200 \cdot 0,6 \cdot 3 = 360 \text{ грн.}$$

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків в кількості 10 примірників:

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 32,5 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 32,5 грн./шт.

$$З_{М2} = 32,5 \cdot 10 = 325 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – 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.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (360 + 325 + 1702)/150 = 16 \text{ грн.}$$

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

$$О_п = З_о \cdot Н_п \cdot 0,01, \quad (7.19)$$

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

$$О_п = 1050 \cdot 15 \cdot 0,01 = 158 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е = 150$ прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 125448 \cdot 3/(15 \cdot 12) = 209 \text{ грн.}$$

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	2	3
1. Основна зарплата виконавців	$З_о$	1050
2. Додаткова зарплата виконавців	$З_д$	105
3. Відрахування на соціальні потреби	$С_{оц}$	254
4. Загальногосподарські витрати	$Г_{осп}$	158
5. Витрати на матеріали	$З_М$	16
6. Освоєння нових операційних систем, мов програмування	$О_п$	158
7. Амортизація основних фондів	$А_м$	209
8. Повна собівартість програмного забезпечення	$С_п$	1950
9. Плановий прибуток	$П_р$	1170
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	3120
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	624
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3744

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

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 1050+105+254+158+16+158+209 = 1950 \text{ грн.}$$

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 60%.

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \qquad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 60 \cdot 1950 = 1170 \text{ грн.}$$

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 пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Вартість програмного рішення взятого для порівняння складе 15000 грн. Результати розрахунків зводимо у таблицю 7.9.

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	4000	3744
Всього капітальних витрат	4000	3744

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	$З_р$	63360	51533
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	1000	936
Всього витрат за рік	$І$	64360	52469

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \quad (7.23)$$

де: $Т_р$ – кількість годин обслуговування сервера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість годин для обслуговування системи зменшилось з 400 годин на рік до 320 годин на рік, тому витрати складуть:

$$З_{р\,баз} = 400 \cdot 120 \cdot 1{,}1 \cdot 1{,}22 = 63360 \; грн,$$

до:

$$З_{р\,нов} = 320 \cdot 120 \cdot 1{,}1 \cdot 1{,}22 = 51533 \; грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Визначити різницю споживання електроенергії при впровадженні систем не має можливості, тому витрати на електроенергію в розрахунку приймаємо рівними нулю.

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	4000	3744	1000	936
Всього відрахувань	-	4000	3744	1000	936

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:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum E_р K_р, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.

$$E_в = (3120 - 1950) \cdot 150 - (0,05 \cdot 1408000 + 0,4 \cdot 113125 + 0,25 \cdot 33190 + 0,1 \cdot 15000) \frac{3}{12} = 144138 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1569315}{(3120 - 1950) \cdot 150 \cdot 12 / 3} = 2,2 \text{ років}.$$

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.	150
2. Повна собівартість розробленої програми	Грн.	1950
3. Ціна розробленої програми	Грн.	3120
4. Плановий прибуток від реалізації розробленої програми	Грн.	1170
5. Рентабельність програмної продукції	%	60
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1569315
7. Загальний прибуток від реалізації програмної продукції	Грн.	175500
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	144138
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	2,2
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3744
11.Величина економічного ефекту у користувача програмної продукції	Грн.	11955
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,1

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б + E_н K_б) - (I_н + E_н K_н), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об’єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (64360+0{,}25\cdot4000)-(52469+0{,}25\cdot3744) = 11955 \text{ грн.}$$

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

$$T_{сп} = \frac{\Delta K}{E_{сп}}, \tag{7.28}$$

$$T_{сп} = \frac{4000-3744}{11955} < 0{,}1 \text{ року}.$$

Як бачимо з розрахунків запропонований варіант є більш економічно доцільним ніж варіант вибраний для порівняння, який на даний момент є лідером продаж на ринку.

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.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона праці та здоров'я у сфері ІТ – це комплекс заходів, які спрямовані на забезпечення безпечних і здорових умов праці для працівників, які використовують інформаційні технології, а також на запобігання травматизму, професійним захворюванням і стресу.

Охорона праці та здоров'я у сфері ІТ включає такі напрями, як:

– Ергономіка – це наука про адаптацію робочого середовища до фізичних і психологічних особливостей людини[2]. Ергономіка вимагає врахування таких факторів, як розміри, форми, кольори, освітлення, шум, температура, вологість, вентиляція, постава, рухи, пози, втома, навантаження на очі та ін. Егономіка допомагає покращити комфорт, продуктивність і задоволення працівників.

– Комп'ютерна безпека – це захист комп'ютерних систем і даних від несанкціонованого доступу, зміни, знищення або блокування. Комп'ютерна безпека вимагає використання антивірусних програм, фаєрволів, паролей, шифрування, резервного копіювання та інших технологій. Комп'ютерна безпека допомагає запобігти крадіжці, шпигунству, шантажу, саботажу та іншим загрозам.

– Соціальна взаємодія – це процес спілкування між людьми у робочому колективі або через мережеві сервіси. Соціальна взаємодія вимагає дотримання правил етикету, поваги, толерантності, співробітництва та конструктивного діалогу. Соціальна взаємодія допомагає покращити настрій, мотивацію, комунікацію та творчий потенціал працівників.

Правила охорони праці і здоров'я для програмістів:

– Регулярно роби перерви в роботі. Вставай із-за столу і розминай м'язи.

– Налаштуй яскравість і контрастність монітору так, щоб не напружувати очі.

– Використовуй ергономічну мишку і клавіатуру, які зручно лягають у руку і не виклкають болю.

– Слідкуй за своєю поставою. Сиди прямо і не нахиляйся до екрану.

– Захищай свій комп'ютер від вірусів, шпигунських програм і хакерів. Оновлюй ативірусне програмне забезпечення і не відкривай підозрілі файли і посилання.

– Не забувай про соціальну взаємодію. Спілкуйся з колегами, друзями і родиною. – Відвідуй заходи, які тебе цікавлять. Не ізолюй себе від світу.

– Люби свою професію, але не забувай про інші сфери життя. Розвивай свої захопленя, хоббі і таланти. Знаходь рівновагу між роботою і відпочинком.

Закон України “Про охорону праці” визначає основні принципи, завдання, права і обов’язки суб’єктів відносин з охорони праці, а також організаційні та правові основи державного управління і контролю за дотриманням законодавства про охорну праці.

Згідно з цим законом, ІТ компанії повинні впроваджувати такі заходи з охорни праці:

– Створювати на підприємстві службу охорони праці або призначати відповідальних осіб, які забезпечують розроблення, реалізацію та контроль за дотриманням заходів з охорони праці.

– Забезпечувати безпечні і нешкідливі умови праці для працівників, використвючи сучасні засоби техніки безпеки, санітарно-гігієнічні умови, засоби клектинго та індивідуального захисту, оптимальні режими праці та відпочинку.

– Проводити атестацію робочих місць на відповідність нормативно-правовим атам з охорони праці та аудит з охорони праці.

– Проводити навчання та інструктаж з питань охорони праці, з надання першої медичної допомоги потерпілим від нещасних випадків і правил поведінки у разі виникнення аварії[5].

– Забезпечувати лікувально-профілактичне обслуговування працюючих, санітарно-побутове обслуговування, пільги і компенсації для працівників, які працюють у важких і шкідливих умовах.

– Нести відповідальність за порушення законодавства про охорону праці та заподіяння шкоди життю і здоров'ю працівників.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом [1]. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– монотонність праці;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Довжина	8,56
Ширина	5,43
Висота	2.9

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,64
Обсяг, V	$м^3$	не менше 20.0	19,25

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працюють 7 людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа приміщення у розрахунку на одного робочого місця програміста відповідає нормативним вимогам як Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров’я

працівників під час роботи з екранними пристроями», так і СанПіН 3.3.2 – 007 – 98.

А об’єм приміщення у розрахунку на одного робочого місця програміста відповідає нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», але дещо нижчий вимог СанПіН 3.3.2 – 007 – 98 (19,25 $м^3$ замість 20 $м^3$). Це цілком прийнятно, враховуючи хронологію та пріоритет опубліковування вищезгаданих нормативних актів.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного дердавного санитарного врача Украини, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виделення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	24-25	55-65	0,1

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосується у холодний період року.

В літню пору застосується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5 – 28 – 2006 р можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути

освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

– бажано, щоб робочий стіл при необхідності можна було регулювати по висоті в межах 680-780 мм, а висота над рівнем підлоги робочої поверхні, на якій працює програміст, повинна складати 720 мм.

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Додаткові рекомендовані заходи: забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення та залулення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

8.5 Розрахунок занулення

Розрахунок занулення складається з трьох частин:

1. Розрахунок на відключаючу спроможність;

2. Визначення максимальної напруги на корпусі обладнання відносно землі при замиканні фази на корпус;

3. Розрахунок робочого і повторного заземлювачів

Початкові дані

1. Потужність електроприладів, які підлягають зануленню:

P = 7 кВт.

2. Довжина магістрального кабеля: L_M=12 м.

3. Довжина розгалудження (від розподільчого щита до електроприладів) : l=25 м.

4. Материал провідників кабеля−алюміній.

5. Лінійна напруга U=380 В.

6. Фазна напруга Uф=220 В.

Розрахунок проводиться відповідно до схеми електромережі, яка зображенана рис. 8.1:

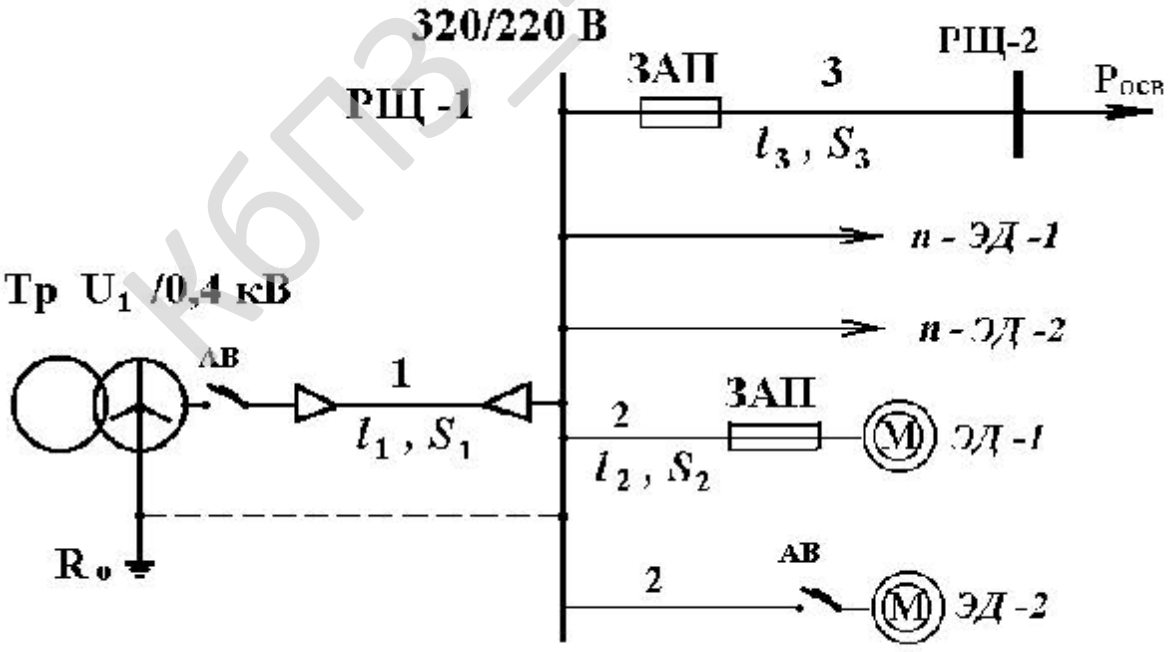

Рисунок 8.1 – Схема електромережі

Тр – трансформатор; РЩ – розподільчий щит; АВ – автоматичний вимикач; ЗАП – запобіжник; 1 – магістральний кабель; 2 – розгалуження до електроприладів.

У результаті розрахунку отримали:

– Номінальна силу струму апарата захисту 15 А.

– Найменше допустиме по умовам спрацьовування захисту значення сили струмукороткого замикання 30 А.

– Площа перерізу магістрального кабелю (провідника) 8 мм^{2}.

– Максимальної напруги на корпусі обладнання відносно землі при замиканні фази на корпус 11,5 В.

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 приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок занулення, рекомендовано заходи з охорони праці.

Список використаних джерел інформації

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

2. Охорона праці. Ч. 2. Занулення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ−сумісного типу / [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака, А. Е. Солових, С. Е. Катеринич] ; Мін-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – 2−ге вид., перероб. та доп. – Кропивницький : ЦНТУ, 2019. – 27 с. URI: http://dspace.kntu.kr.ua/jspui/handle/123456789/8769

3. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення 19.10.23).

4. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. – 19 с. [Електронний ресурс]. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12240 (дата звернення 19.09.23).

5. НПАОП 40.1-1.21-98. Правила безпечної експлуатації електроустановок споживачів – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0093-98#Text (дата звернення 19.09.23).

6. 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 (дата звернення 19.09.23).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи структурного кодування даних у комп’ютерних мережах автоматизованих систем управління.

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 структурного кодування даних у комп’ютерних мережах автоматизованих систем управління.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем структурного кодування даних у комп’ютерних мережах автоматизованих систем управління.

– Досліджена система структурного кодування даних у комп’ютерних мережах автоматизованих систем управління.

– На основі отриманих результатів досліджень створена програмна реалізація системи структурного кодування даних у комп’ютерних мережах автоматизованих систем управління.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання структурного кодування даних у комп’ютерних мережах автоматизованих систем управління.

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.

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм AES.

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 11955 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,1 рік.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Скрипник Д.А. Дослідження та програмна реалізація системи структурного кодування даних у комп'ютерних мережах автоматизованих систем управління // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. *Бабак В.П., Василенко В.С., Юдін О.К.* Оцінка впливу параметрів каналу зв'язку на відносну швидкість обміну даними // Вісник НАУ. – 2005.- №4 (26). – С. 3-7

3. *Черниш Л.Г., Юдін О.К.* Загальний підхід до побудови систем віддаленого адміністрування, автоматичного оновлення і супроводу програмних продуктів // Право і безпека. Т.4.- К.: НУВС, 2005. -№4. – С. 185-188

4. *Поляков П.Ф., Юдін О.К., Яковенко О.Л.* Аналіз процедур несанкціонованого доступу до інформаційних ресурсів відкритих систем згідно з еталонною моделлю ISO/OSI // Телекомунікаційні системи та мережі на залізничному транспорті. – Х., 2006.- Випуск №78. – С. 130-138

5. Vijay Kumar Velu. Mastering Kali Linux for Advanced Penetration Testing. Packt Publishing Ltd. 2022. 573 p.

6. Josh Armitage. Cloud Native Security Cookbook. O'Reilly Media. 2022. 516 p.

7. Massimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 p.

8. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

9. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

10. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

11. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

12. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

13. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

14. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

15. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

16. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

17. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

18. 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,

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

20. 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.

21. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE*

International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418

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

23. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

24. 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.*

25. 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.

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

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

28. 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.

29. 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.

30. 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.

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

32. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

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

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

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

40. 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.

41. 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.

42. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

43. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced*

Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18 - 21 September 2019. P.713-718.

44. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

45. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

46. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

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

48. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

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

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

51. 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.

52. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи структурного кодування даних у комп’ютерних мережах автоматизованих систем управління.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 36-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи структурного кодування даних у комп’ютерних мережах автоматизованих систем управління.

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;

– 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 структурного кодування даних у комп'ютерних мережах автоматизованих систем управління;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Visual C#.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 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 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута шкідливі і небезпечні фактори при роботі з комп’ютером.

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 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 14.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Дрєєва Г.М.

Дослідження та програмна реалізація
системи структурного кодування даних у комп’ютерних мережах
автоматизованих систем управління

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 56

Літера: РП

Кропивницький – 2023 року

Файл MainForm.cs - головне вікно програми

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

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

        #region Windows Form Designer generated code

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

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

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

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

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

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

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

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

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

        }

        #endregion

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

Файл StructureCodeDecoder.cs - декодер структурного кодування даних

```
using System;
using System.Threading;

namespace RecoveryDisk
{
    /// <summary>
    /// Клас декодера структурного кодування даних
    /// </summary>
    public class StructureCodeDecoder : StructureCodeBase
    {
        #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 StructureCodeDecoder()
        {
            // 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 StructureCodeDecoder(int dataCount, int eccCount, int[] volList)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, volList,
(int)StructureCodeType.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 структурного кодування даних (по
типу матриці)</param>
        public StructureCodeDecoder(int dataCount, int eccCount, int[] volList,
int codecType)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, volList, codecType);
```

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

        #endregion Construction & Destruction

        #region Public Operations

        /// <summary>
        /// Установка конфігурації декодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="volList">Список порядкових номерів наявних
томів</param>
        /// <param name="codecType">Тип кодека кодера структурного кодування
даних (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)StructureCodeType.Dispersal)
            {
                maxVolCount = (int)StructureCodeConst.MaxVolCountDisp;

            } else
            {
                maxVolCount = (int)StructureCodeConst.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.eStructureCodeType)
                    )
                {
                    this.mainConfigChanged = true;
                }

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

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

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

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

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

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

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

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

                this.configIsOK = true;

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

            return this.configIsOK;
        }

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

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

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

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

                    data[i] = mulSum;
```

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

            return true;
        }

        #endregion Public Operations

        #region Private Operations

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

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

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

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

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

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

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

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

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

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

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

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

                    return false;
                }

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

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

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

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

                        int idx = i_n + j;

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

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

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

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

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

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

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

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

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

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

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

                    return false;
                }
            }

            return true;
        }

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

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

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

        /// <summary>
```

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

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

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

                return;
            }

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

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

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

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

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

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

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

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

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

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

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

                    return;
```

```
            }
        }

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

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

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

                return;
            }
        }

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

        // Якщо основна конфігурація змінилася...
        if (this.mainConfigChanged)
        {
            if (this.eStructureCodeType == (int)StructureCodeType.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.eStructureCodeType == (int)StructureCodeType.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 (OnStructureCodeMatrixFormingFinish != null)
            {
                //...повідомляємо, що екземпляр класу готовий до роботи
                OnStructureCodeMatrixFormingFinish();
            }

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

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

        #endregion Private Operations

        #region Public Properties

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

                } else
                {
                    return null;
                }
            }
        }

        #endregion Public Properties
    }
}
```

Файл StructureCodeEncoder.cs - кодер структурного кодування даних

```
using System;
using System.Threading;

namespace RecoveryDisk
{
    /// <summary>
    /// Клас кодера структурного кодування даних
    /// </summary>
    public class StructureCodeEncoder : StructureCodeBase
    {
        #region Construction & Destruction

        /// <summary>
        /// Конструктор кодера за замовчуванням
        /// </summary>
        public StructureCodeEncoder()
        {
            // 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 StructureCodeEncoder(int dataCount, int eccCount)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, (int)StructureCodeType.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 структурного кодування даних (по
типу матриці)</param>
        public StructureCodeEncoder(int dataCount, int eccCount, int codecType)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, codecType);

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

        #endregion Construction & Destruction

        #region Public Operations

        /// <summary>
        /// Установка конфігурації кодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="codecType">Тип кодека кодера структурного кодування
даних (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)StructureCodeType.Dispersal)
            {
                maxVolCount = (int)StructureCodeConst.MaxVolCountDisp;

            } else
            {
                maxVolCount = (int)StructureCodeConst.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.eStructureCodeType)
                    )
                {
                    this.mainConfigChanged = true;
                }

                // Зберігаємо конфігурацію
                this.n = dataCount;
                this.m = eccCount;
                this.eStructureCodeType = 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.eStructureCodeType ==
(int)StructureCodeType.Alternative)
                {
                    this.iterOfFirstStage = m;

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

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

                this.configIsOK = true;

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

```
            }

            return this.configIsOK;
        }

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

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

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

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

                ecc[i] = mulSum;
            }

            return true;
        }

        #endregion Public Operations

        #region Private Operations

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

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

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

                        return;
                    }
```

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

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

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

                        return;
                    }
                }

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

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

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

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

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

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

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

                        // Активуємо індикатор актуального стану змінних-членів
```

```
                        this.finished = true;

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

                        return;
                    }
                }

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

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

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

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

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

        #endregion Private Operations
    }
}
```

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

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

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

        #region Windows Form Designer generated code

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

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

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

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

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

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

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

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

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

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

        }

        #endregion

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

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

Файл FileAnalyzer.cs - контроль цілісності даних

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

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

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

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

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

        #endregion Delegates

        #region Public Properties & Data

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

                } else
                {
                    return false;
                }
            }
        }

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

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

        } else
        {
            return false;
        }
    }
}

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

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

        } else
        {
            return false;
        }
    }
}

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

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

        } else
        {
            return null;
        }
    }
}

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

/// <summary>
```

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

                } else
                {
                    return false;
                }
            }
        }

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

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

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

                            break;
                        }

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

                            break;
                        }

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

                            break;
                        }

                        case 3:
                        {
```

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

                            break;
                        }

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

                            break;
                        }
                    }

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

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

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

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

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

        #endregion Public Properties & Data

        #region Data

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

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

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

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

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

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

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

        /// <summary>
        /// Тип кодера структурного кодування даних (по типу використовуваної
матриці кодування)
        /// </summary>
        private int codecType;

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

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

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

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

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

        #endregion Data

        #region Construction & Destruction

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

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

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

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

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

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

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

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

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

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

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

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

        #endregion Construction & Destruction

        #region Public Operations

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

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

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

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

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

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

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

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

                return false;
            }

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

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

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

                return false;
            }

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

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

            // Зберігаємо тип кодера структурного кодування даних (по типу
використовуваної матриці кодування)
            this.codecType = codecType;

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

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

                // Повертаємо результат обробки
                return this.processedOK;
            }

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

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

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

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

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

            // Повідомляємо, що все нормально
            return true;
        }

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

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

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

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

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

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

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

                // Установлюємо подію завершення обробки
```

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

                return false;
            }

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

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

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

                return false;
            }

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

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

            // Зберігаємо тип кодека кодера структурного кодування даних (по
типу використовуваної матриці кодування)
            this.codecType = codecType;

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

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

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

                // Повертаємо результат обробки
                return this.processedOK;
            }

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

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

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

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

            // Повідомляємо, що все нормально
            return true;
        }

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

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

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

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

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

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

        #endregion Public Operations

        #region Private Operations

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

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

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

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

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

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

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

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

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

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

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

                            continue;
                        }

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

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

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

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

                            return;
                        }
```

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

                    } // while(true)

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

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

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

                    return;
                }

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

                    } else
                    {
                        break;
                    }
                }

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

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

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

                    return;
                }

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

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

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

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

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

                    return;
                }
            }

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

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

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

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

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

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

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

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

            // Індекс у масиві томів
            int volListIdx = 0;

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

            // Лічильник кількості ушкоджених основних томів
```

```
            int dataVolMissCount = 0;

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

            // Ім'я файлу для обробки
            String fileName;

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

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

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

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

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

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

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

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

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

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

                                    continue;
                                }

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

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

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

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

                                    return;
                                }

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

                            } // while(true)

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

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

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

                            return;
                        }

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

                            } else
                            {
```

```
                                break;
                            }
                        }

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

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

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

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

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

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

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

                        return;
                    }
                }

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

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

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

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

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

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

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

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

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

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

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

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

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

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

                                continue;
                            }

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

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

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

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

                                return;
                            }

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

                        } // while(true)

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

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

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

                        return;
                    }

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

                        } else
                        {
                            break;
                        }
                    }

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

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

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

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

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

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

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

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

                        return;
                    }
                }

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

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

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

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

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

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

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

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

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

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

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

                return;
            }

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

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

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

                return;
            }

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

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

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

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

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

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

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

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

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

        #endregion Private Operations
    }
}
```

Файл About.cs - вікно довідки про програму

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

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

        #region Windows Form Designer generated code

        /// <summary>
        /// Необхідний метод для підтримки розробника - не модифікується
        /// зміст цього метода використовується редактором коду.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.developersListBoxdevelopersListBox = new
System.Windows.Forms.ListBox();
            this.StructureCodeIconTimer = new
System.Windows.Forms.Timer(this.components);
            this.okButtonXP = new PinkieControls.ButtonXP();
            this.SuspendLayout();
            //
            // developersListBoxdevelopersListBox
            //
            this.developersListBoxdevelopersListBox.BackColor =
System.Drawing.SystemColors.Control;
            this.developersListBoxdevelopersListBox.BorderStyle =
System.Windows.Forms.BorderStyle.None;
            this.developersListBoxdevelopersListBox.FormattingEnabled = true;
            this.developersListBoxdevelopersListBox.Items.AddRange(new object[]
{
            "МАГІСТЕРСЬКА РОБОТА",
            "",
            "На тему:",
            "",
            " Дослідження та програмна реалізація системи структурного кодування
даних у комп’ютерних мережах автоматизованих систем управління",
            "",
            "",
            "Керівник: Дрєєва Г.М.",
            "",
            "Розробив: студент Скрипник Дмитро Анатолійович",
            "                  гр. КІ-22МЗ",
            "",
            "м. Кропивницький 2023"});
            this.developersListBoxdevelopersListBox.Location = new
System.Drawing.Point(11, 6);
            this.developersListBoxdevelopersListBox.Name =
"developersListBoxdevelopersListBox";
```

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

        }

        #endregion

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

    }
}
```