Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Токар А.М.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Буравченко К.О.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *125 “*Кібербезпека*”*

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

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

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

Токарю Артему Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 57-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Буравченко К.О.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Токар А.М.
(прізвище та ініціали)

АНОТАЦІЯ

Токар А.М. Програмне забезпечення системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

Метою розробки є програмне забезпечення системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

Результат роботи – програмна реалізація системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих 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 з ОС Windows 10/11.

Програму розроблено в середовищі Visual C#.

Ключові слова: кібербезпека, синтез структурно-блокових кодів

ABSTRACT

Tokar A.M. Software for the cybersecurity system of synthesis of structural-block codes for specialized automated control systems. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the cybersecurity system of synthesis of structural-block codes for specialized automated control systems.

The purpose of the development is the software for the cybersecurity system of synthesis of structural-block codes for specialized automated control systems.

The result of the work is the software implementation of the cybersecurity system of synthesis of structural-block codes for specialized 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 are provided.

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

The program was developed in Visual C#.

Keywords: cybersecurity, synthesis of structural block codes

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АСУ	–	Автоматизована система управління
БГ	–	Базова група
СБК	–	Структурно-блоковий код

ВСТУП

Актуальність теми. Постійне розширення галузі застосування автоматизованих інформаційних систем (АІС), побудова складних систем централізованого та розподіленого керування, в яких цифрові пристрої і системи є їх невід'ємною частиною, збільшення обсягів переданої та обробленої інформації призводять до зростання вимог по забезпеченню надійності, відмовостійкості й швидкодії. Вирішення даної проблеми можливе завдяки підвищенню надійності елементної бази, введенню різних видів надмірності, оптимізації логічних структур, поліпшенню технічного обслуговування на основі розробки ефективних методів прогнозування. У даний час розвиток відмовостійких автоматизованих систем управління (АСУ) перебуває під особливим контролем держав і відноситься до розряду стратегічно важливих і актуальних проблем наукових розробок, зокрема для створення перспективних систем управління озброєнням і технікою. систем управління авіаційним і залізничним транспортом, енергетикою, хімічною промисловістю і т. д.

Основними автоматизованими процесами служать збирання, передача, прийом, зберігання й обробка інформації. Збільшення обсягів оброблюваних даних і складності розв'язуваних системами управління задач обумовлює вдосконалювання й розробку принципово нових відмовостійких засобів переробки інформації. Значний внесок у вирішення проблеми створення надійних АСУ за допомогою інформаційної надмірності, а саме кодування інформації, зроблено науковцями Дж. Фон Нейманом, К. Шенноном, Дж. Поуєном, С. Виноградовим, М.О. Гавриловим, О.П. Стаховим, Є.І. Брюховічем, Ю.Г. Дадаєвим, О.В. Ткаченко та ін. Аналіз існуючих кодових систем свідчить, що їх властивості залежать від рівня надмірності. У даний час, частка контрольного устаткування в АІС і АСУ постійно збільшується. Кількісний перехід до якісного стрибка можливий тільки при сполученні робочих і контролюючих процесів. Дану ідеологію повною мірою дозволяють реалізувати структурно-блокові коди,

створення яких має об'єктивний характер і обумовлене тенденцією підвищення імунітету спеціалізованих АСУ до помилок і несправностей.

Нині відсутня загальна теорія щодо позиційних, надлишкових систем числення, не досліджені проблеми підвищення надійності, відмовостійкості й швидкодії на основі кодів із природною надлишковістю, не визначені оцінки ефективності їх контролюючих властивостей, тощо...

Таким чином, у роботі пропонується новий підхід стосовно вирішення проблеми забезпечення надійності відмовостійкості й швидкодії спеціалізованих АСУ, котрий у даний час недостатньо досліджений та полягає у спільному використанні надмірного кодування інформації і апаратної надмірності схемотехнічної реалізації на основі використання структурно-блокових кодів.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

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

– Огляд існуючих систем синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

– Дослідження системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

– Програмна реалізація системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації методів синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління. Проведемо аналіз сучасного стану й тенденцій розвитку інформаційних систем, визначимо АІС які забезпечують достовірну обробку оперативної інформації в реальному часі. Визначимо такі властивості як надійність, відмовостійкість та швидкодія. Проаналізуємо сучасний стан й тенденції розвитку елементної бази, принципів, методів і засобів забезпечення надійності й відмовостійкості спеціалізованих АСУ.

На основі обраних часткових і загальних якісних характеристик кодів і пристроїв визначаються комплексні показники якості функціонування АСУ:

1. Вірогідність функціонування – імовірність відповідності вихідних даних правильному результату:

$$D = 1 - \sum_{i=1}^{N}\sum_{l=1}^{n} \beta_i \cdot P_i^l \left(1 - \sum_{j=0}^{\xi} k_{Doj}^l\right). \quad (1.1)$$

2. Вірогідність виправлення – імовірність правильного виправлення:

$$D_u = 1 - \sum_{i=1}^{N}\sum_{l=1}^{n} \beta_i \cdot P_i^l \left(1 - \sum_{j=0}^{\xi} k_{Dkj}^l\right). \quad (1.2)$$

3. Імовірність правильної роботи – імовірність одержання правильного результату:

$$P_p = 1 - \sum_{i=1}^{N}\sum_{l=1}^{n} \beta_i \cdot P_i^l \left(1 - \sum_{j=0}^{\xi} (k_{Doj}^l - k_{Dkj}^l)\right). \quad (1.3)$$

4. Імовірність безвідмовної роботи:

$$P_{\text{В}} = 1 - \sum_{i=1}^{N}\sum_{l=1}^{n} \beta_i \cdot P_i^l \left(1 - \sum_{j=0}^{\xi} (\lambda_1 \cdot k_{Doj}^l + \lambda_2 \cdot k_{Dkj}^l)\right). \quad (1.4)$$

5. Коефіцієнт швидкодії:

$$K_B = \frac{T_P^*}{(T_P + T_K)}. \quad (1.5)$$

У комплексних показниках позначені:

N – загальна кількість пристроїв;

n – довжина кодової комбінації;

β – коефіцієнт зміни ймовірності виникнення помилки в розряді, що залежить від структури коду, несиметричності каналів і статико-динамічного режиму роботи пристрою;

P – імовірність відмови роботи пристрою, що реалізує традиційну безнадлишкову систему числення;

l – кратність помилки;

ξ – кількість методів контролю інформації;

k_{Do} і k_{Dk} – коефіцієнти вірогідності контролю й корекції інформації;

λ_1 і λ_2 – інтенсивності відмов і перебоїв функціональних елементів;

T_P^* і T_P – час рішення задач управління вихідною безнадлишковою системою і системою контролюючої помилки, виходячи з максимальної кількості послідовних операцій інформаційного потоку.

Виходячи із залежностей (1.1) – (1.5), розв'язання задачі синтезу припускає розробку нових інформаційних і схемотехнічних побудов дискретних пристроїв з метою досягнення екстремальних показників відмовостійкості й швидкодії.

1.2 Область застосування

Областю застосування системи є автоматизовані системи управління. Сутність проблеми підвищення надійності й відмовостійкості полягає в досягненні ряду граничних умов:

$$P, \lambda_1, \lambda_2, \beta, \to 0, \quad (1.6)$$

$$k_{Do}, k_{Dk} \to 1, \tag{1.7}$$

$$\zeta, l \to \max. \tag{1.8}$$

Сутність проблеми підвищення швидкодії полягає у виконанні умов

$$T_P, T_K \to 0, \tag{1.9}$$

$$T_P < T_p^* \to 0. \tag{1.10}$$

Сукупність співвідношень (6) – (10) являє собою постановку наукової проблеми досліджень у формалізованому вигляді.

Наукова проблема синтезу високонадійних, відмовостійких АСУ концептуально буде вирішуватися в процесі логічного синтезу. Методологічна основа синтезу пристроїв АСУ з природною надлишковістю полягає у введенні надлишковості на найбільш ранньому етапі – етапі алгоритмічного синтезу, що передбачає формалізацію алгоритму роботи автомата.

Розглядаються теоретичні основи синтезу високонадійних відмовостійких дискретних пристроїв, пропонується залежно від обраної групи систем числення застосування однієї або кількох стратегій самокорекції.

Узагальнення вимог до синтезу кодів і пристроїв на їхній основі дозволило сформулювати загальні підходи до створення необхідних кодів і пристроїв:

1. Визначення задач, що розв'язуються проектованою АСУ.
2. Ставлення вимог щодо контролю помилок і швидкодії.
3. Вибір або виведення коду, що забезпечує введення надлишковості й визначає показники D, D_u, P_p, P_B, і K_B, для цього визначаються можливості кодування й контролю обчислювальної інформації, оцінюється адекватність контрольованих помилок реальній моделі, вплив зміни статико-динамічного режиму роботи пристроїв на інтенсивності потоків відмов і перебоїв.
4. Узгодження класу розв'язуваних задач і вимоги щодо контролю інформації й швидкодії, розробляються методи контролю, корекції, переробки інформації, формалізуються алгоритми роботи пристроїв.
5. Для підвищення контролюючих можливостей визначаються декілька кодових структур у рамках обраної рекурентної послідовності розрахунку

вагових коефіцієнтів систем числення, на основі яких розробляються додаткові методи корекції, які ґрунтуються на реконфігурації кодів, систем числення або апаратури.

6. Для підвищення швидкодії розробляються методи високошвидкісного мікроконвеєрного виконання операцій, синтезуються алгоритми сполучення виконання операцій обробки й контролю.

7. Моделюються математичні моделі пристроїв, визначаються їхні складності, проводиться розрахунок показників D, D_u, P_p, P_B, і K_B, робиться висновок про виконання вимог або зміну коду в пункті 3.

8. Синтезується набір природно-надлишкових модулів, що забезпечує синтез АСУ.

Пункти 3-7 є центральними й забезпечують розробку інформаційної бази досягнення умов (1.6) – (1.10), на основі яких створюється методологічна й схемотехнічна база створення АСУ, адаптивної до класу розв'язуваних задач і застосовуваної технології виробництва.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

HandyBits EasyCrypto

HandyBits EasyCrypto – це невеликий додаток, що допоможе захистити конфіденційну інформацію шляхом кодування файлів і папок.

Використовувати програму досить просто. Вам потрібно буде імпортувати необхідні файли й папки за допомогою файлового браузера, потім увести пароль і нажати клавішу “Шифрування” (розшифровка відбувається в такий же спосіб).

У списку ви зможете переглянути шлях, розмір і тип кожного файлу. Крім того ви зможете створити що саморозпаковується або Zip архів, вибрати довжину ключа шифрування, увести привітальне повідомлення, відправити на пошту й багато чого іншого.

У налаштуваннях програми ви зможете управляти списками паролів, змінювати тло й мову інтерфейсу, відключити підказки, а також налаштувати інші параметри.

Особливості програми для кодування файлів і папок:

– Можливість створити зашифровані й стислі архіви, які саморозпаковуються.

– Ви можете шифрувати й дешифрувати файли й папки прямо з Windows Explorer.

– Безпечне видалення вихідного коду після шифрування.

– Сильні алгоритми шифрування.

– Багатомовний інтерфейс.

DigiSecret

Основні функції DigiSecret – створення закодованих архівів і передача цих архівів між користувачами. Інтерфейс програми, по суті, нагадує різні архіватори: для створення архіву потрібно перетягнути потрібні файли в основне вікно програми й, вибравши відповідний пункт меню, створити закодований архів.

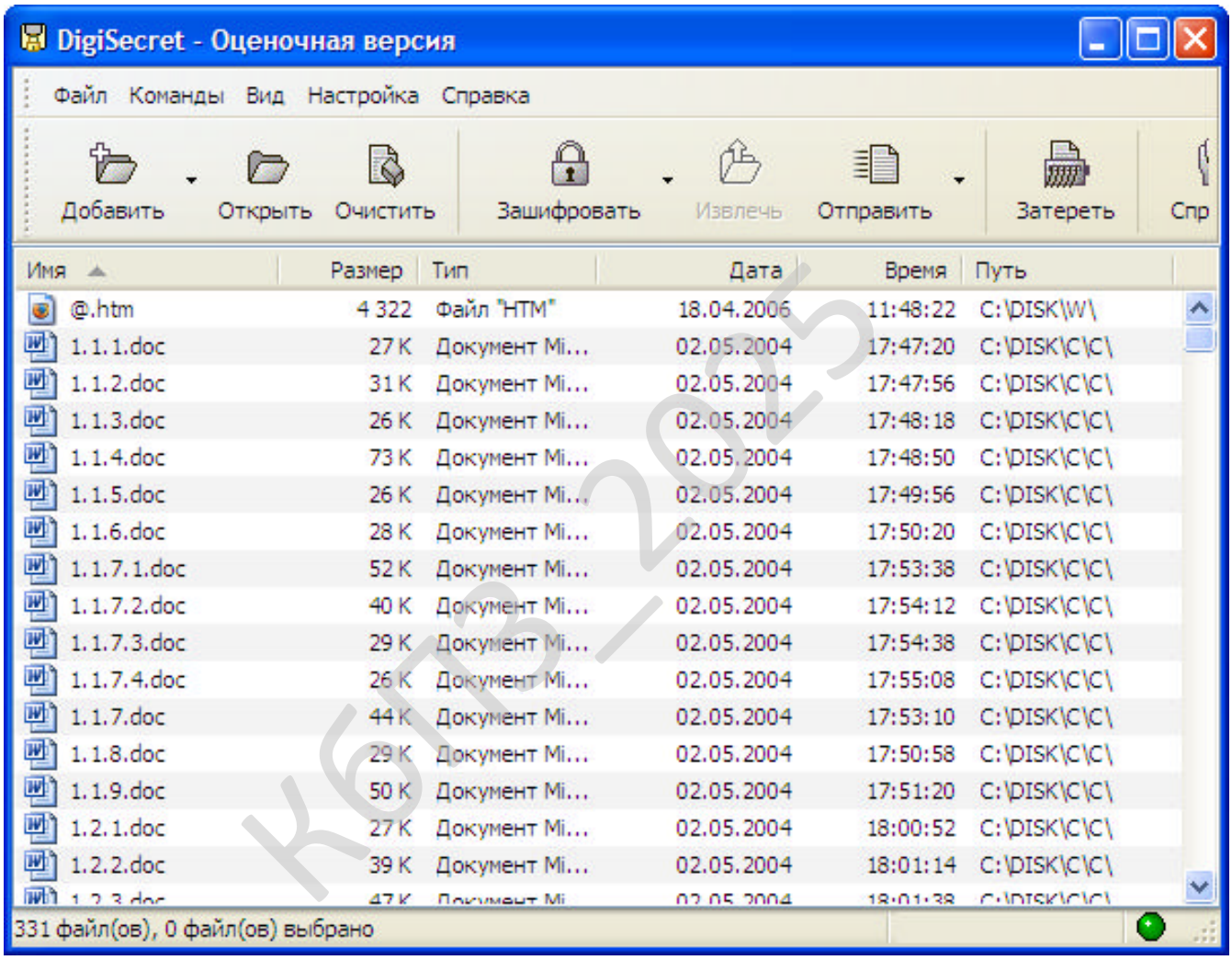

Рисунок 2.1 – Головне вікно програми

Зрозуміло, при створенні архіву використовується механізм компресії. Виробник затверджує, що використовуваний в DigiSecret механізм компресії надзвичайно ефективний, однак на практиці виявляється, що навіть убудований в Windows XP механізм компресії ZIP справляється із цим завданням краще.

Як бачимо, відставання DigiSecret від WinRAR на тестовому завданні склало 0,5 МБ, або майже на 7%. Втім, компресія – не головне завдання програми, і її рівень достатній для середнього користувача.

При створенні архіву можна вибрати одну з дев'яти ступенів стиску відповідно до бажаної швидкості архівації, або відключити компресію зовсім, і тоді програма просто зашифрує вміст файлів. DigiSecret уміє створювати й SFX (які саморозпаковуються) архіви, що напевно придасться в тих випадках, коли виникає необхідність передати інформацію людині, що не має своєї копії цієї програми. У створеному архіві зберігається структура папок, хоча в основному вікні програми всі файли «звалені в купу». У програмі передбачена можливість видалення вихідних файлів після створення архіву, причому, як і личить рішенню, що служить для забезпечення безпеки даних, DigiSecret пропонує на вибір кілька варіантів механізму їхнього затирання – просте видалення, а також затирання в 1, 4, 9 і 35 проходів. Затирання можна зробити й без створення архіву, вибравши відповідний пункт меню.

DigiSecret інтегрується в Провідник Windows, надаючи можливість виконати найбільше часто використовувані операції за допомогою контекстного меню – створити звичайний або архів, що саморозпаковується, безпечно видалити файл або папку, або відправити потрібний об'єкт у закодованому виді іншому користувачеві DigiSecret або по електронній пошті.

Програма пропонує на вибір кілька алгоритмів кодування – CAST (128-бітний ключ), Blowfish (448-бітний ключ), Twofish (256-бітний ключ) і Rijndael (також відомий як AES, 256-бітний ключ).

На питання про кращий вибір алгоритму сама компанія однозначно відповісти не може – на думку авторів програми, всі запропоновані варіанти гарні. Алгоритм CAST-128 стійкий до лінійного й диференціальному криптаналізу, і може бути зламаний тільки методом повного перебору. Twofish і Blowfish, розроблені відомим фахівцем з безпеки Брюсом Шнайєром (Bruce Schneier), також є стійкими й швидкими шифрами, про уразливість яких нічого не

відомо. Rijndael є переможцем конкурсу на AES (Advanced Encryption Standard) і в 2000 році став стандартом у США. Якщо припустити, що ці алгоритми можуть бути зламані тільки методом повного перебору, то ключі довжиною 128 біт зможуть протистояти атакам протягом багатьох десятиліть, а ключі довжиною 448 біт виключають навіть теоретичну можливість атаки. Одним словом, компанія залишає вибір варіанта за користувачем. Програма не використовує майстер-ключ, розшифровка можлива тільки ключем, використаним при шифруванні. Слід зазначити, що розмір створюваного архівного файлу залишається незмінним незалежно від обраного алгоритму кодування.

DigiSecret пропонує користувачам на вибір два варіанти безпечної передачі інформації – іншим користувачам DigiSecret за допомогою убудованих у програму засобів або по електронній пошті у вигляді кодованого файлу.

Для передачі файлів засобами програми необхідно помістити потрібні об'єкти в основне вікно й вибрати в меню «Команди» пункт «Відправити через DigiSecret». Що відкрилося після цього вікно налаштувань передачі дозволяє вибрати потрібного адресата, указавши його IP-адресу, а також алгоритм кодування й пароль. Для зручності вибору часто використовуваних адресатів у програмі передбачена проста адресна книга.

При використанні для передачі інформації електронної пошти програма створює кодований архів (пропонуючи попередньо вибрати його налаштування) і відкриває поштовий клієнт із новим листом і вкладеним у нього архівом. Користувачеві залишається тільки вибрати потрібного адресата й при необхідності дописати повідомлення.

До речі, про повідомлення. Програма має убудований Центр кодування повідомлень, що дозволяє зашифрувати текстове повідомлення за допомогою одного з алгоритмів і передати його по електронній пошті. Для його розшифровки адресат, зрозуміло, повинен знати пароль.

Відмінність версій Lite і Pro

У версії Pro доступні наступні функції, відсутні в Lite:

– Центр кодування повідомлень, що дозволяє кодувати й відправляти текстові повідомлення по електронній пошті.

– Можливість обмінюватися файлами через TCP/IP з іншими користувачами DigiSecret.

– Можливість змінювати SFX-архіви шляхом додавання заголовка й гіперпосилання, що буде виконувати задану команду.

Всі інші можливості доступні в обох версіях. В DigiSecret Lite, як і у версії Pro, їсти можливість кодувати й декодувати архіви, створювати SFX-файли, розділяти файли й т.д.

Переваги:

– Русифікований інтерфейс.

– Можливість використання замість архіватора.

– Створення архівів, що саморозпаковуються.

– Інтеграція з поштовими клієнтами.

– Інтеграція із Провідником Windows.

– Передача інформації із принципу p2p.

– Можливість вибору алгоритму кодування.

– Ефективне затирання файлів.

Недоліки:

– Відсутність деревоподібного подання структури папок у головному вікні програми.

– Низька, у порівнянні з популярними архіваторами, ступінь компресії.

Утиліта DigiSecret, володіючи щодо невисокою вартістю, здатна замінити архіватор, надаючи при цьому можливість значно підвищити безпека передачі й зберігання даних. Простий локалізований інтерфейс, зручність використання, інтеграція в поштові клієнти й оболонку Windows – все це сприяє «швидкому звиканню» до використання програми, а виходить, користувач може виключити можливість перехоплення інформації, не прикладаючи особливих зусиль.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Синтаксис Visual C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C++ або Java з легкістю взнають синтаксис із фігурними дужками, характерний для мови Visual C#. Розроблювачі, що знають кожну із цих 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 Win32, у мові Visual C# можна використовувати процес, що називається "Interop". Процес Interop дозволяє програмам на Visual C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C++. Мова Visual C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення.

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

Архітектура платформи .NET Framework

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

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

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

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

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Сформулюємо вимоги до інформаційної надлишковості, що вводиться, розглянемо способи її введення, одним із яких є введення структурної надлишковості на рівні кодів подання інформації.

Визначення 1. **Структурно-блоковим кодом (СБК)** називається код, для всіх комбінацій якого існує правило чергування блоків символів.

Визначення 2. **Базовою групою (БГ)** СБК називається мінімальний набір комбінацій символів, допустимих у коді.

Структура коду і його структурні властивості зосереджені в структурі блока символів.

Визначити структурні властивості блока символів можна через набори комбінацій символів, що входять в них.

На основі поняття БГ доведені такі теореми:

Теорема 1: Будь-який СБК має БГ.

Теорема 2: Будь-який код, що не має контрольних розрядів, при $n \to \infty$ й $M(n) \to \infty$ має БГ, де n – довжина кодової комбінації, $M(n)$ – потужність коду.

Визначено основні вимоги до БГ і правил її формального опису, це дозволило спростити процес алгоритмізації синтезу кодів і розробку математичного апарата аналізу СБК.

Виходячи з вимог, структуру БГ у загальному вигляді можна представити так:

$$\begin{cases} \underbrace{1...1}_{v_{11}}\underbrace{2...2}_{v_{12}}\underbrace{3...3}_{v_{13}}...\underbrace{r-1...r-1}_{v_{1n}}\underbrace{0...0}_{v_{10}} \\ \underbrace{1...1}_{v_{21}}\underbrace{2...2}_{v_{22}}\underbrace{3...3}_{v_{23}}...\underbrace{r-1...r-1}_{v_{2n}}\underbrace{0...0}_{v_{20}} \\ \underbrace{1...1}_{v_{31}}\underbrace{2...2}_{v_{32}}\underbrace{3...3}_{v_{33}}...\underbrace{r-1...r-1}_{v_{3n}}\underbrace{0...0}_{v_{30}} \\ \quad ..... \\ \underbrace{1...1}_{v_{m1}}\underbrace{2...2}_{v_{m2}}\underbrace{3...3}_{v_{m3}}...\underbrace{r-1...r-1}_{v_{mn}}\underbrace{0...0}_{v_{m0}} \end{cases} \quad (3.1)$$

де r – основа системи числення, v_{ij} – кількість символів j-го виду в i-му слові БГ коду.

Проведемо аналіз методів синтезу СК.

Для усунення виявлених недоліків і забезпечення можливості створення автоматизованої системи проектування на основі БГ запропонуємо нові підходи до синтезу СБК:

– на основі перестановок слів БГ, з додатковим сортуванням по зростанню кодів цифр;

– послідовним вибором комбінацій символів, що задовольняє вимогам БГ.

Виходячи з цього, алгоритм методу синтезу СБК на основі БГ включає:

– вибір і обґрунтування структури коду;

– визначення БГ СБК;

– синтез коду в явному вигляді на основі запропонованих підходів.

Практична реалізація методів показала перевагу другого підходу через менші вимоги до пам'яті, що дозволило збільшити потужність синтезованих кодів і розширити кількість кодових підмножин.

Раніше досліджені й уперше отримані кодові підмножини були класифіковані за такими ознаками:

– кількість символів алфавіту: двійкові СБК; трійкові СБК; четвіркові СБК;...; r-кові СБК;

– довжина серії однойменних символів: СБК з необмеженою й обмеженою довжиною серії однойменних символів;

– кількість видів символів у словах БГ: СБК рівномірної, нерівномірної та

змішаної структури;

– обмеження на кількість однойменних символів у коді: СБК з виключенням і без виключення кількості однойменних символів, з виключенням і без виключення нулів;

– обмеження на якість застосовуваних символів: СБК з виключенням і без виключення символів;

– розмір блока однойменних символів: СБК мінімальної (оптимальної) форми, пакетної форми, пакетно-змінної форми;

– наявність вагових коефіцієнтів розрядів: вагозначущі СБК; невагозначущі, з багатомірними ваговими рядами;

– виявлення помилок: такі, що не виявляють помилки; такі, що виявляють помилки; такі, що гарантовано виявляють помилки заданої кратності;

– виправлення помилок: такі, що не виправляють помилки; такі, що виправляють помилки; такі, що гарантовано виправляють помилки заданої кратності.

Виділені в класифікації підмножини були визначені на основі структури БГ (3.1) і формалізовані, що дозволило розпочати їх аналіз.

3.2 Розробка структурної схеми

Визначимо залежності для моделювання характеристик СБК, синтезованих на основі БГ за класифікованими підмножинами. Наведемо основні результати дослідження СБК мінімальної форми.

Під функцією СБК мається на увазі набір матриць і послідовність їхнього об'єднання при структурно-матричному методі синтезу:

$$f_n^* = z \cdot (f_1^* + f_2^* + ... + f_{n-d}^*),$$

де:

f_m^* – матриця m-розрядних кодових слів, у старшому розряді яких перебуває ненульовий символ;

z – кількість ненульових символів алфавіту коду;

d – максимальна кількість символів у слові БГ.

Інформативність коду визначається:

$$M_n = \sum_{i=1}^{n} M_i^* = z \cdot M_{n-d} + M_{n-1},$$

де:

$M_n^* = z \cdot M_{n-d}^* + M_{n-1}^* = z \cdot M_{n-d}$ – кількість рядків у матриці f_m^* .

Кількість символів i-го виду в n-розрядному коді визначається:

$$K_n^i = K_n^{i*} + K_{n-1}^i,$$

де:

$K_n^{i*} = z \cdot K_{n-d}^i + M_{n-d}$ – кількість символів i-го виду в матриці f_m^* , $i \in \{1,...,z\}$,

при цьому $K_n^{1*} = K_n^{2*} = K_n^{3*} = ... = K_n^{z*}$ й $K_n^1 = K_n^2 = K_n^3 = ... = K_n^z$.

Будь-яке число A може бути представлене в СБК як:

$$A = \sum_{i=1}^{n} (B_i^0 + B_i^{**} \cdot \psi_i),$$

де:

$$B_i^0 = \begin{cases} 0, & при\ \psi_i = 0 \\ B_i^*, & при\ \psi_i \neq 0 \end{cases}$$ – постійна складова

B_n^{**} – змінна складова вагового коефіцієнта,

$B_n^* = B_n^m - m \cdot B_n^{**} = z \cdot B_{n-3}^* + B_{n-1}^* = B_n^1 - B_n^{**}$,

$$B_n^{**} = M_{n-d} = z \cdot B_{n-d}^{**} + B_{n-1}^{**} = B_n^m - B_n^{m-1} = B_{n-2}^{**} + B_{n-2}^*,\ m \in \{2, z-1\}.$$

Для вагових коефіцієнтів справедливі такі залежності:

$$B_n^1 = M_{i-1} = z \cdot B_{n-2}^1 + B_{n-1}^1 = B_{n-1}^z + B_{n-d}^1,$$

$$B_n^m = B_n^{m-1} + B_{n-d+1}^1 = B_n^1 + (m-1) \cdot B_{n-d+1}^1,$$

$$B_n^i = z \cdot B_{n-3}^i + B_{n-1}^i,$$

де $i \in \{1,...,z\}$.

Блок визначення підкласу кодів за вимогами до структури коду

↓

Блок визначення системи обмежень даного підкласу, що дозволяє формально описати сукупність досліджуваних кодів

↓

Блок визначення найбільш істотних параметрів формального опису

↓

Блок визначення рекурентних послідовностей для розрахунку основних показників структурно-блокових кодів (СБК)

1. Обрані параметри визначаються як мінімально допустимі, і на їхній основі визначається в явному вигляді базовою групою (БГ) заданого коду.
2. Застосувавши метод перебору для синтезу структурно-блокових кодів (СБК) на основі базової групи (БГ), одержимо коди цифр у явному вигляді.
3. На основі отриманих кодів визначаються початкові значення основних показників структурно-блокових кодів (СБК) (M_n^*, M_n, K_n^{i*}, K_n^i, B_n^i, B_n^*, B_n^{**}).
4. Визначаються рекурентні послідовності для розрахунку основних показників структурно-блокових кодів (СБК), і доводиться їхня правильність.
5. Один з параметрів збільшується на одиницю й повторюються пункти 1-4.

↓

Блок визначення рекурентних послідовності в загальному вигляді

6. На основі двох варіантів рекурентних послідовностей визначаються рекурентні послідовності в загальному вигляді й доводиться їхня правильність.
7. Пункти 5-6 повторюються для урахування зміни всіх обраних параметрів.

↓

Блок оптимізації набору загальних залежностей для розрахунку необхідних параметрів структурно-блокових кодів (СБК), на основі кількості виконуваних операцій

↓

Блок розрахунку параметрів і вибору необхідного структурно-блокового коду (СБК)

Рисунок 3.1 – Структурна схема системи

На рисунку 3.1 зображена структурна схема системи. Логіка її роботи визначається наступними міркуваннями. Взявши за основу методику синтезу загальних залежностей для СБК, був сформульований метод дослідження СБК без їх повного синтезу:

1. За вимогами до структури коду визначається підклас кодів.

2. Визначається система обмежень даного підкласу, що дозволяє формально описати сукупність досліджуваних кодів.

3. Визначаються найбільш істотні параметри формального опису.

4. Обрані параметри визначаються як мінімально допустимі, і на їхній основі визначається в явному вигляді БГ заданого коду.

5. Застосувавши метод перебору для синтезу СБК на основі БГ, одержимо коди цифр у явному вигляді.

6. На основі отриманих кодів визначаються початкові значення основних показників СБК (M_n^*, M_n, K_n^{i*}, K_n^i, B_n^i, B_n^*, B_n^{**}).

7. Визначаються рекурентні послідовності для розрахунку основних показників СБК, і доводиться їхня правильність.

8. Один з параметрів збільшується на одиницю й повторюються пункти 4 – 7.

9. На основі двох варіантів рекурентних послідовностей визначаються рекурентні послідовності в загальному вигляді й доводиться їхня правильність.

10. Пункти 8 – 9 повторюються для урахування зміни всіх обраних параметрів.

11. На основі кількості виконуваних операцій оптимізується набір загальних залежностей для розрахунку необхідних параметрів СБК.

12. Проводиться розрахунок параметрів і вибирається необхідний СБК.

Для спрощення реалізації запропонованого методу наведені основні результати досліджень СБК пакетної форми з необмеженою серією нулів, оптимальної форми й пакетної форми з обмеженою серією символів, обрані набори залежностей, що дозволяють проводити розрахунок параметрів даних

кодових сукупностей з найменшою трудомісткістю.

Для визначення контролюючих і коригувальних можливостей СБК на основі БГ синтезовані таблиці контрольованих і коректованих переходів символів:

$$T_{ij}^{o} = \begin{vmatrix} \alpha_{11}^{o}\ \alpha_{12}^{o}\ \alpha_{13}^{o}\ \ldots\ \alpha_{1z}^{o} \\ \alpha_{21}^{o}\ \alpha_{22}^{o}\ \alpha_{23}^{o}\ \ldots\ \alpha_{2z}^{o} \\ \alpha_{31}^{o}\ \alpha_{32}^{o}\ \alpha_{33}^{o}\ \ldots\ \alpha_{3z}^{o} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \alpha_{z1}^{o}\ \alpha_{z2}^{o}\ \alpha_{z3}^{o}\ \ldots\ \alpha_{zz}^{o} \end{vmatrix} \qquad T_{ij}^{k} = \begin{vmatrix} \alpha_{11}^{k}\ \alpha_{12}^{k}\ \alpha_{13}^{k}\ \ldots\ \alpha_{1z}^{k} \\ \alpha_{21}^{k}\ \alpha_{22}^{k}\ \alpha_{23}^{k}\ \ldots\ \alpha_{2z}^{k} \\ \alpha_{31}^{k}\ \alpha_{32}^{k}\ \alpha_{33}^{k}\ \ldots\ \alpha_{3z}^{k} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \alpha_{z1}^{k}\ \alpha_{z2}^{k}\ \alpha_{z3}^{k}\ \ldots\ \alpha_{zz}^{k} \end{vmatrix},$$

де:

T_{ij}^{o} і T_{ij}^{k} – таблиці контрольованих і коректованих переходів j-го символу i-го слова БГ,

$a_{xy}^{0} \in \{0.1\}$ – визначає виявлення (1) або пропуск помилки (0) при помилці переходу j-го символу в i-му слові БГ зі значення x в y,

$a_{xy}^{k} \in \{0.1\}$ – визначає виправлення (1) або пропуск помилки (0) при помилці переходу j-го символу в i-му слові БГ зі значення x в y.

Кількість помилок, що виявляються словами БГ коду, визначаються:

$$v = \sum_{i=1}^{m_x} \sum_{j=1}^{m_y} v_{ij},$$

де:

$v_{ij} = \sum_{x=1}^{z} \sum_{y=1}^{z} a_{xy}^{o}$ – кількість помилкових переходів j-го символу в i-му слові БГ, що виявляються;

m_x – кількість слів у БГ коді;

m_y – максимальна кількість розрядів у слові БГ коду.

Кількість помилок, коректованих у словах БГ коду, визначається:

$$h = \sum_{i=1}^{m_x} \sum_{j=1}^{m_y} h_{ij},$$

де:

$h_{ij} = \sum_{x=1}^{z}\sum_{y=1}^{z} a_{xy}^{k}$ – кількість коректованих помилкових переходів j-го символу в i-му слові БГ;

m_x – кількість слів у БГ коді;

m_y – максимальна кількість розрядів у слові БГ коду.

Взявши за основу формалізовані алгоритми синтезу СБК, були отримані залежності для розрахунку контролюючих і коригувальних можливостей кодів на основі рекурентних послідовностей. Наприклад, для СБК мінімальної форми й пакетної форми з необмеженими серіями нулів кількість помилок, що виявляються, визначається:

$$K_n^o = z \cdot K_{n-d}^o + K_{n-1}^o + v \cdot M_{n-d} + v_0 \cdot M_{n-1},$$

де:

K_n^o – кількість помилок в n-розрядному коді, що виявляються,

v_0 – кількість помилок у старшому нульовому розряді коду, що виявляються.

Кількість коректованих помилок для даних кодів визначається:

$$K_n^k = z \cdot K_{n-d}^k + K_{n-1}^k + h \cdot M_{n-d} + h_0 \cdot M_{n-1},$$

де:

K_n^k – кількість коректованих помилок у n-розрядному коді,

h_0 – кількість коректованих помилок у старшому нульовому розряді коду.

Для СБК оптимальної форми й пакетної форми з обмеженими серіями нулів кількість помилок, що виявляються, визначається:

$$K_n^o = z \cdot \sum_{i=\upsilon_{\min}+1}^{\upsilon_{\max}+1} K_i^o + v \cdot \sum_{i=\upsilon_{\min}+1}^{\upsilon_{\max}+1} M_i + v_0 \cdot M_{n-1}.$$

Кількість коректованих помилок для даних кодів визначається:

$$K_n^k = z \cdot \sum_{i=\upsilon_{\min}+1}^{\upsilon_{\max}+1} K_i^k + h \cdot \sum_{i=\upsilon_{\min}+1}^{\upsilon_{\max}+1} M_i + h_0 \cdot M_{n-1}.$$

Наводяться вирази для розрахунку ймовірностей виявлення й виправлення помилок. Наводяться зворотні послідовності для розрахунку кількості помилок, що виявляються і виправляються, залежно від їхньої кратності.

На прикладі операції додавання розглянемо можливість обробки обчислювальної інформації в АСУ, представленої СБК. Розробимо метод моделювання правил виконання операції додавання, що включає таке:

1. На основі обраної БГ СБК визначаються початкові значення вагових рядів, при цьому кількість рядів повинна дорівнювати z, а кількість початкових значень кожного ряду – не менше $z+1$.

2. Розрахунок наступних значень вагових коефіцієнтів на основі рекурентної послідовності $B_n^i = B_{n-1}^i + z \cdot B_{n-d}^i$.

3. Перевірка рівності $B_n^{y_1} + B_n^{y_2} = B_n^{x_1} + B_{n-1}^{x_2} + B_{n-2}^{x_3} + \ldots + B_{n-k}^{x_{k-1}}$ при повному перебиранні y_i і x_i.

4. Повторна перевірка отриманого правила при $n^* > n$ для виключення випадкової рівності сум вагових коефіцієнтів.

5. Підрахунок кількості ненульових доданків у правій частині отриманої рівності.

6. Прийняття рішення про включення отриманої рівності або заміну наявної при меншій складності правої частини рівності.

Для СБК мінімальної форми, отриманих на основі даного методу, правила виконання операції додавання були узагальнені й подані в такому формалізованому вигляді:

$$\begin{cases} 0+0=0 \\ b_n^m+0=0+b_n^m=b_n^m \\ b_n^m+b_n^p=b_n^p+b_n^m=\begin{cases} b_n^{m+p-1}+b_{n-1}^z+b_{n-d}^1; & при (m+p)<(z+2) \\ b_{n+1}^1+b_n^{m+p-z-1}; & при (m+p)>(z+1) \end{cases} \end{cases}$$

Метод моделювання правил виконання операції нормалізації аналогічний методу моделювання правил виконання операції додавання, за винятком пункту 3, у якому перевіряється тільки рівність $B_n^{y_1}+B_{n-1}^{y_2}+B_{n-2}^{y_3}+\ldots+B_{n-k}^{y_{k-1}}=B_n^{x_1}+B_{n-1}^{x_2}+B_{n-2}^{x_3}+\ldots+B_{n-k}^{x_{k-1}}$.

Уведена інформаційна надлишковість СБК у деяких випадках повинна дозволити розбити виконання операції додавання на 2 етапи:

– виконання операції додавання з одержанням результату в забороненій формі, при реалізації якої функція переносу відсутня;

– нормалізація результату виконання операції, тобто зведення його до дозволеного вигляду.

Виходячи з цього, операції додавання, віднімання, множення будуть виконуватися на основі конвеєра, що забезпечує підвищення швидкодії обчислювального процесу. Для одержання можливості досягнення даного необхідного результату був проведений аналіз вагових рядів, отриманих при синтезі кодів на основі перебирання при зміні їхніх початкових значень. Результати проведення обчислювального експерименту показали, що для двійкового СБК, заданого зворотною послідовністю $B_n=B_{n-1}+B_{n-5}$ при початкових значеннях, частина яких подана в табл. 1, справедлива рівність $B_n=B_{n-1}+B_{n-5}=B_{n-2}+B_{n-3}$.

Для вагових коефіцієнтів операція додавання буде виконуватися на основі виразів:

$$\begin{cases} 0+0=0; \\ b_n+0=0+b_n=b_n; \\ b_n+b_n=b_n+b_{n-1}+b_{n-5}=b_{n-2}+b_{n-3}. \end{cases}$$

Даний метод додавання виключає поширення сигналів переносу за межі локальної 5-ї розрядної групи. Вагові коефіцієнти забезпечують існування СБК із заданими властивостями, при цьому вагові коефіцієнти в рядку 4 є більш переважними, тому що збільшують діапазон подання чисел на 10 – 20%.

У процесі розробки методу виконання операції додавання для СБК, заданого $БГ = \begin{cases} 10 \\ 20 \\ 0 \end{cases}$, було відзначено, що подвоєний ваговий коефіцієнт цифри «1» на одиницю відрізняється від наступного вагового коефіцієнта даної цифри. Це послужило основою створення методу додавання на основі віднімання й додання одиниць:

$$\begin{cases} 0+0=0; \\ b_n^1+0=0+b_n^1=b_n^1; \\ b_n^2+0=0+b_n^2=b_n^2; \\ b_n^1+b_n^1=b_{n+1}^1+(-1)^n; \\ b_n^2+b_n^2=b_{n+1}^2; \\ b_n^1+b_n^2=b_n^2+b_n^1=b_{n+1}^1+b_{n-1}^1+(-1)^n. \end{cases}$$

Віднімання й додання одиниць дозволяє спростити виконання операції нормалізації. Реалізація методу додавання на основі віднімання й додання одиниць, припускає конвеєрне виконання додавання на основі таких етапів:

– виконання додавання з одержанням результату в забороненій формі;

– часткова нормалізація результату додавання на основі віднімання й додання одиниць;

– корекція результату додавання по різниці виділених одиниць;

– нормалізація остаточного результату додавання.

Розглянемо можливість реалізації отриманих теоретичних результатів по дослідженню СБК у двійковому базисі. Практичний перехід до даного базису можливий за двома напрямками: синтез двійкових СБК і кодування СБК у двійковому базисі.

Визначимо залежності для моделювання характеристик двійкових СБК із обмеженою серією символів і необмеженою серією нулів як окремий випадок раніше отриманих загальних залежностей. Результати аналізу кодованих двійкових СБК показали, що одержувані кодові сукупності як за структурою, так і за характеристиками частково успадковують вихідний код і спосіб кодування, але мають нові якісні й кількісні характеристики. На основі уточнених аналітичних залежностей розроблений матрично-аналітичний метод синтезу двійкових СБК, визначені переваги даного методу синтезу.

Контроль помилок у СБК полягає в перевірці відповідності кодової комбінації системі обмежень і зводиться до одного з двох підходів:

– перевірка відповідності довжини груп однойменних символів;

– перевірка відповідності груп символів, послідовності символів у словах БГ.

Реалізація першого підходу приводить до контролю коду на основі функцій:

$$F = F_1 \cap F_2 \cap ... \cap F_h \cap ... \cap F_n;$$
$$F_h = F_h^{1\to 0} \cup F_h^{0\to 1},$$
$$F_h^{1\to 0} = F_h^{*1\to 0} \cup F_{h-1}^{*1\to 0} \cup ... \cup F_{h-\nu_{i1}}^{*1\to 0},$$
$$F_h^{*1\to 0} = \bar{Q}_h\, Q_{h+1} ... Q_{h+\nu_{11}} \cup \bar{Q}_h\, Q_{h+1} ... Q_{h+\nu_{21}} \cup ... \cup \bar{Q}_h\, Q_{h+1} ... Q_{h+\nu_{m1}},$$
$$F_h^{0\to 1} = F_h^{*0\to 1} \cup F_{h-1}^{*0\to 1} \cup ... \cup F_{h-\nu_{i0}}^{*0\to 1}$$
$$F_h^{*0\to 1} = \bar{Q}_h\, \bar{Q}_{h+1} ... \bar{Q}_{h+\nu_{10}-1} \cup \bar{Q}_h \bar{Q}_{h+1} ... \bar{Q}_{h+\nu_{20}-1} \cup ... \cup \bar{Q}_h\, \bar{Q}_{h+1} ... \bar{Q}_{h+\nu_{m0}-1}.$$

де:

$F = 0$ при наявності помилки;

Q_h – функція h-го входу;

$1\to 0$ і $0\to 1$ – помилки переходу одиниці в нуль і нуля в одиницю відповідно.

Реалізація другого підходу приводить до контролю коду на основі функцій:

$$F = F_1 \cap F_2 \cap ... \cap F_h \cap ... \cap F;$$
$$F_h = .F_h^* \cup F_{h-1}^* \cup ... \cup F_{h-\nu_{\min}}^*,$$
$$F_h^* = \bar{Q}_h\, Q_{h+1} ... Q_{h+\nu_{11}} \bar{Q}_{h-1} \bar{Q}_{h-2} ... \bar{Q}_{h-\nu_{10}} \cup \bar{Q}_h\, Q_{h+1} ... Q_{h+\nu_{21}} \bar{Q}_{h-1} \bar{Q}_{h-2} ... \bar{Q}_{h-\nu_{20}} \cup$$
$$\cup ... \cup \bar{Q}_h\, Q_{h+1} ... Q_{h+\nu_{m1}} \bar{Q}_{h-1} \bar{Q}_{h-2} ... \bar{Q}_{h-\nu_{m0}}.$$

де:

$\nu_{\min} = \min(\nu_{i1} + \nu_{i0})$, $i \in \{1, m\}$.

Помилки найефективніше виявляються й виправляються СБК пакетної форми. Перетворення оптимальних і мінімальних ДСБК у ДСБК пакетної форми при довжині пакета, що дорівнює m, виконується на основі виразу:

$$C_h = Q_{h+m} \cup Q_{h+m-1} \cup ... \cup Q_{h+1} \cup Q_h,$$

де:

h, m – натуральні числа;

C_h – функція h-го виходу пристрою;

Q_h – функція h-го входу.

Зворотне перетворення кодів виконується на основі виразу: $C_h = Q_h Q_{h-1} Q_{h-2} ... Q_{h-m}$.

Взаємне перетворення кодів виконується на основі виразу:

$$C_h = Q_h Q_{h-1} ... Q_{h-n} \cup Q_{h+1} Q_h ... Q_{h-n+1} \cup Q_{h+2} Q_{h+1} ... Q_{h-n+2} \cup ... \cup Q_{h+m} Q_{h+m-1} ... Q_{n+m-n}.$$

Контроль помилок виконується на основі виразу:

$F = F_1 \cup F_2 \cup ... \cup F_h \cup ... \cup F_n$

$F_h = \overline{Q_h}\, Q_{h+1}\, \overline{Q_{h+2}} \cup Q_h\, Q_{h+1}\, Q_{h+2}$ для $m = 2$,

$F_h = \overline{Q_h}\, Q_{h+1}\, Q_{h+2}\, \overline{Q_{h+3}} \cup \overline{Q_h}\, Q_{h+1}\, \overline{Q_{h+2}} \cup Q_h\, Q_{h+1}\, Q_{h+2} Q_{h+3}$ для $m = 3$,

$F_h = \overline{Q_h}\, Q_{h+1}\, Q_{h+2} Q_{h+3}\, \overline{Q_{h+4}} \cup \overline{Q_h}\, Q_{h+1}\, Q_{h+2}\, \overline{Q_{h+3}} \cup \overline{Q_h}\, Q_{h+1}\, \overline{Q_{h+2}} Q_{h+3} \cup Q_h\, Q_{h+1}\, Q_{h+2} Q_{h+3} Q_{h+4}$ для $m = 4$,

...

Виправлення помилок виконується на основі виразу:

$$\Phi = \Phi_1 \cup \Phi_2 \cup ... \cup \Phi_h \cup ... \cup \Phi_n$$

$$\Phi_h = Q_{h-1}\,\overline{Q_h}\,Q_{h+1} \quad \text{для} \quad m = 3,$$

$$\Phi_h = Q_{h-1}\,\overline{Q_h}\,Q_{h+1}Q_{h+2} \cup Q_{h-2}\,Q_{h-1}\,\overline{Q_h}\,Q_{h+1} \quad \text{для} \quad m = 4,$$

$$\Phi_h = Q_{h-1}\,\overline{Q_h}\,Q_{h+1}Q_{h+2}Q_{h+3} \cup Q_{h-2}\,Q_{h-1}\,\overline{Q_h}\,Q_{h+1}Q_{h+2} \cup$$

$$\cup Q_{h-3}Q_{h-2}\,Q_{h-1}\,\overline{Q_h}\,Q_{h+1} \quad \text{для} \quad m = 5,$$

де:

Φ_h – відкоректований вихідний сигнал в h-m розряді.

На основі запропонованих моделей синтезовані функціональні вузли комбінаційного типу.

Забезпечення самокорекції проводиться відповідно до стратегій:

– виявлення помилки – виправлення – контроль результату на наступному кроці;

– виправлення – контроль результату на наступному кроці;

– виявлення непоправної помилки – реконфігурація – повторне виконання операції – декодування результату з виправленням помилки – контроль результату.

Розглядаються такі методології реконфігурації:

– реконфігурація коду без зміни вагових коефіцієнтів розрядів і правил виконання операції додавання;

– реконфігурація системи числення зі зміною коду числа, вагових коефіцієнтів розрядів і правил виконання операцій додавання;

– часткова реконфігурація системи числення зі зміною коду числа без зміни вагових коефіцієнтів і правил виконання операції додавання. Даний підхід найповніше використовує інформаційну надлишковість СБК.

Розглядаються такі методи корекції на основі реконфігурації:

– СБК пакетної форми шляхом збільшення пакетів символів:

1) виявлення помилок $F_h = \overline{Q_h} Q_{h+1} \overline{Q_{h+2}} \cup Q_h Q_{h+1} Q_{h+2} = (Q_h \cup Q_{h+2}) Q_{h+1}$;

2) перетворення коду $C_h = (Q_h \cup Q_{h+1} Q_{h+2}) \cap F$;

3) повторне виконання операції в перетвореному коді;

4) декодування з виправленням помилок $\Phi_h = \Phi_h^1 \cup \Phi_h^2 \cup \Phi_h^3 \cup \Phi_h^4$,

де:

$\Phi_h^1 = F(h+1)(Q(h+1)Q(h)Q(h-1)Q(h-2) \vee Q(h+2)Q(h+1)Q(h)Q(h-1))$;

$\Phi_h^2 = F(h)(\overline{Q(h+1)}\ \overline{Q(h)}Q(h-1)Q(h-2) \vee \overline{Q(h+2)}\ \overline{Q(h+1)}Q(h)Q(h-1))$;

$\Phi_h^3 = F(h-1)(\overline{Q(h+1)}Q(h)\overline{Q(h-1)}Q(h-2) \vee \overline{Q(h+2)}\ Q(h+1)\overline{Q(h)}Q(h-1))$;

$\Phi_h^4 = F(h-2)(\overline{Q(h+1)}\ Q(h)Q(h-1)Q(h-2) \vee \overline{Q(h+2)}\ Q(h+1)Q(h)Q(h-1))$.

– СБК оптимальної форми:

1) виявлення помилок $F = F_1 \cup F_2 \cup ... \cup F_h \cup ... \cup F_n$,

де:

$F_h = F_h^{0\to1} + F_h^{1\to0}$; $F_h^{0\to1} = \bar{Q}_h \cap \bar{Q}_{h-1} \cap \bar{Q}_{h-2} \cap ... \cap \bar{Q}_{h-a}$;

$F_h^{1\to0} = Q_h \cap (Q_{h-1} \cup Q_{h-2} \cup ... \cup Q_{h-b})$;

2) перетворення коду $C_h = (Q_h \cup Q_{h+1} \cup Q_{h+2}) \cap F$;

3) повторне виконання операції в перетвореному коді;

4) декодування з виправленням помилок $\Phi_h = Q_{h-1} \cap Q_{h-2}$.

– СБК пакетної форми шляхом зменшення пакетів символів:

1) виявлення помилок $F = F_1 \cup F_2 \cup ... \cup F_h \cup ... \cup F_n$,

де:

$F_h = F_h^1 + F_h^2$; $F_h^1 = \bar{Q}_{h+1}Q_h\bar{Q}_{h-1}\bar{Q}_{h-2}$; $F_h^2 = \bar{Q}_{h+1}Q_hQ_{h-1}Q_{h-2}$;

2) перетворення коду $C_h = (Q_h \cap Q_{h-1}) \cap F$;

3) повторне виконання операції в перетвореному коді;

4) декодування з виправленням помилок. $\Phi_h = Q_{h+1} \cup Q_{h+2}$.

Тимчасове резервування, викликане повторним виконанням операції при перебоях і відмовах, забезпечує виконання поставленої перед АСУ задачі.

3.3 Розробка функціональної схеми

Розглянемо можливість зменшення апаратної складності реалізації АСУ шляхом введення інформаційної надлишковості СБК на прикладі обчислювального пристрою. Активна інформаційна надлишковість називається негативною, якщо виконується нерівність $K_a > K_u$; позитивною, якщо виконується нерівність $K_a \le K_u$; оптимальною, якщо виконується нерівність $K_a \le 1$. Розроблено метод синтезу кодів з оптимальною інформаційною надлишковістю, що дозволяє без зміни швидкодії зменшити відносну складність обчислювального пристрою. В основу створення методу покладене припущення, що форми подання інформації, які потребують найменшої апаратної складності пристрою, забезпечать найбільш прості алгоритми його функціонування.

Для забезпечення функціонування обчислювального пристрою обмежимося такими операціями: додавання натуральних чисел; віднімання натуральних чисел; віднімання цілих чисел зі знаком; розподіл натуральних чисел; порівняння натуральних чисел.

Якщо обмеження формалізувати виразом $f_{2k+1} = x_{2\cdot k+1} \cap x_{2\cdot k}$, де f_i – функція обмежень при побудові i-го розряду коду; x_i – значення i-го розряду коду, буде синтезована активно-надлишкова двійково-трійкова система числення зі значеннями ваг розрядів 1, 2, 2, 6, 6, 18, 18, 54, 54, 162, 162, 486....

Ваги розрядів даної системи числення визначаються:

$$\begin{cases} B_1 = 1 \\ B_2 = 2 \\ B_{2n} = 3 \cdot B_{2n-1} \\ B_{2n+1} = \cdot B_{2n} \end{cases}.$$

Контроль помилок буде виконуватися відповідно до виразу $F = F_1 \cup F_2 \cup F_3 \cup ... \cup F_k \cup ... \cup F_m$,

де:

$F_k = F_{2p+1} = x_{2p+1} \cup \overline{x}_{2p}$; $m = n/2$;

F_k – функція контролю помилки в k-й дворозрядній групі коду числа;

x_i – значення i-го розряду коду числа.

Були формалізовані правила виконання операції додавання. Один з варіантів моделі суматора для даної системи числення буде мати вигляд:

$$\begin{cases} S_{2m} = A_{2m}\overline{B}_{2m}P_m \cup \overline{A}_{2m}B_{2m+1}P_m \cup A_{2m+1}B_{2m+1}P_m \cup A_{2m}\overline{A}_{2m+1}\overline{B}_{2m+1}P_m \cup \\ \cup \overline{A}_{2m}\overline{B}_{2m}B_{2m+1}P_m \cup \overline{A}_{2m}\overline{A}_{2m+1}B_{2m}\overline{B}_{2m+1}P_m \\ S_{2m+1} = A_{2m+1}B_{2m+1} \cup A_{2m}\overline{A}_{2m+1}\overline{B}_{2m} \cup \overline{A}_{2m}\overline{B}_{2m+1}P_m \cup \overline{A}_{2m}B_{2m}P_m \cup \\ \cup A_{2m+1}B_{2m}P_m \cup A_{2m+1}\overline{B}_{2m}\overline{P}_m \cup A_{2m}B_{2m+1}P_m \cup A_{2m}\overline{A}_{2m+1}\overline{B}_{2m+1}P_m \\ P_{m+1} = A_{2m}B_{2m} \cup A_{2m}B_{2m+1} \cup A_{2m+1}P_m \cup B_{2m+1}P_m \cup A_{2m}B_{2m}P_m \end{cases},$$

де:

A_{2m} і A_{2m+1} – молодший і старший розряди дворозрядної групи першого складу відповідно,

B_{2m} і B_{2m+1} – молодший і старший розряди дворозрядної групи другого складу відповідно,

S_{2m} і S_{2m+1} – молодший і старший розряди дворозрядної групи результату відповідно,

P_m і P_{m+1} – сигнал переносу з попередньої дворозрядної групи і в наступну дворозрядну групу відповідно.

Проведені розрахунки показали, що дана система числення може бути віднесена до оптимальних, активно-надлишкових систем числення, тому що забезпечує спрощення суматора на 16,6% відносно безнадлишкового при однакових швидкодіях.

Якщо обмеження формалізувати виразом, буде синтезована активно-надлишкова двійково-шісткова система числення зі значеннями ваг розрядів 1, 2, 2, 6, 12, 12, 36, 72, 72, 216, 432, 432....

Ваги розрядів даної системи числення визначаються:

$$\begin{cases} b_{3m} = 6^m; \\ b_{3m+1} = 2 \cdot 6^m; \\ b_{3m+2} = 2 \cdot 6^m, \end{cases}$$

де $m = n/3$ – порядковий номер трирозрядної групи розрядів.

Проведені розрахунки показали, що при однакових швидкодіях суматорів двійково-шісткова система числення забезпечує зменшення складності пристрою на 32% при виявленні 25% помилок. Крім того, уведена надлишковість забезпечує збільшення швидкодії на 21%.

Для підвищення оперативності й вірогідності управління АСУ а також для її сумісності із застосовуваними системами узагальнюються результати дослідження кодів з постійною кількістю одиниць і на їхній основі синтезуються $2R$ системи числення. Взявши за основу коди з постійною кількістю одиниць при $a = 1$, можна представити в явному вигляді вагові коефіцієнти розрядів коду (де a – кількість одиничних символів у слові БГ коду):

- двійкова система числення з постійною кількістю одиниць:

$0 \cdot 2^0, 1 \cdot 2^0, 0 \cdot 2^1, 1 \cdot 2^1, 0 \cdot 2^2, 1 \cdot 2^2, \ldots, 0 \cdot 2^n, 1 \cdot 2^n;$

- двійково-трійкова система числення з постійною кількістю одиниць:

$0 \cdot 3^0, 1 \cdot 3^0, 2 \cdot 3^0, 0 \cdot 3^1, 1 \cdot 3^1, 2 \cdot 3^1, \ldots, 0 \cdot 3^n, 1 \cdot 3^n, 2 \cdot 3^n;$

- двійково-четвіркова система числення з постійною кількістю одиниць:

$0 \cdot 4^0, 1 \cdot 4^0, 2 \cdot 4^0, 3 \cdot 4^0, \ldots, 0 \cdot 4^n, 1 \cdot 4^n, 2 \cdot 4^n, 3 \cdot 4^n$....

Будь-яке ціле число X може бути представлене $2R$ кодом у вигляді:

$$X = \pm \sum_{j=0}^{n-1} \sum_{i=0}^{r-1} x_{i+j} i r^i, \quad x \in [0; r-1].$$

Оскільки ваговий ряд $2R$ системи числення можна умовно розбити на блоки по *r*-розрядів у кожному блоці й представлене будь-яке число може мати тільки одну одиницю в кожному блоці розрядів, пристрій виявлення помилок буде мати вигляд:

$$F = \overline{F_1} \vee \overline{F_2} \vee \ldots \vee \overline{F_K} \vee \ldots \vee \overline{F_N}, \qquad k \in [1,N],$$

де:

$$F_K = C_1 \overline{C_2} \overline{C_3} \ldots \overline{C_r} \vee \overline{C_1} C_2 \overline{C_3} \ldots \overline{C_r} \vee \ldots \vee \overline{C_1} \overline{C_2} \overline{C_3} \ldots \overline{C_{r-1}} C_r,$$

C_i – j-й вхід пристрою контролю $j \in [1,r]$.

Імовірність виявлення помилки даним пристроєм визначається як:

$$P_{ОБ}(n) = 1 - 2^{-r} \left(\frac{r}{2} \right)^p.$$

Результати аналізу свідчать про те, що ймовірність виявлення помилок у системах числення з постійною кількістю одиниць незначно вища, ніж у кодах з постійною кількістю одиниць.

Розробимо методику синтезу математичних моделей суматорів для кодів з постійною кількістю одиниць, що включає:

– вибір і обґрунтування вихідної системи числення;

– синтез перших r значень чисел у явному вигляді;

– побудову таблиці істинності роботи суматора;

– побудову математичної моделі суматора по таблиці істинності;

– визначення складності реалізації синтезованого суматора.

Дана методика не містить у собі етап мінімізації традиційними методами. Це дозволяє спростити процеси розробки арифметичних пристроїв підвищеної надійності в $2r$ численнях, а отримані математичні залежності визначення складності суматорів дозволяють найбільш повно провести аналіз пристроїв з погляду надійності ще на першому етапі проектування. Розроблено алгоритми прискореного додавання й множення, у яких кожний розряд результату є функцією тільки суміжних розрядів у межах групи, при цьому час виконання операції не залежить від довжини операндів. Результати моделювання логічних пристроїв показали, що оптимальними будуть моделі при поданні команд, даних і результатів у $2R$ системі числення, тому що мінімальна складність і відсутність інверсій дозволяє спростити складність при підвищенні до 30% швидкодії.

Для забезпечення процесу дослідження запропонованих кодів і спеціалізованих пристроїв АСУ на їх основі, опираючись на отримані наукові і практичні результати даної роботи, створені інструментальні засоби синтезу

СБК, моделювання параметрів кодів, моделювання дискретних пристроїв та розрахунку D, D_u, P_p, P_B, і K_B. Функціональна схема системи представлена на рис. 3.2.

Рисунок 3.2 – Функціональна схема системи

3.4 Розробка діаграми процесів

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

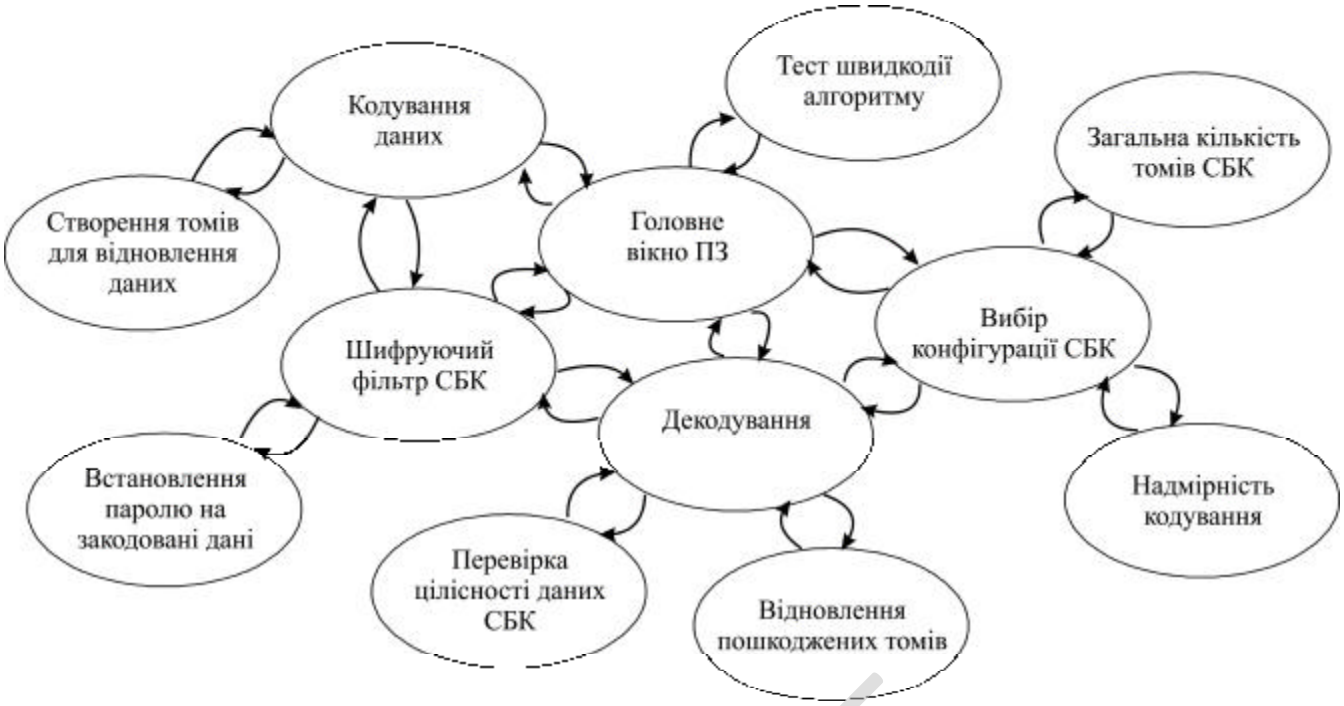

Рисунок 3.3 – Діаграма взаємодії процесів

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 синтезу структурно-блокових кодів для спеціалізованих 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.

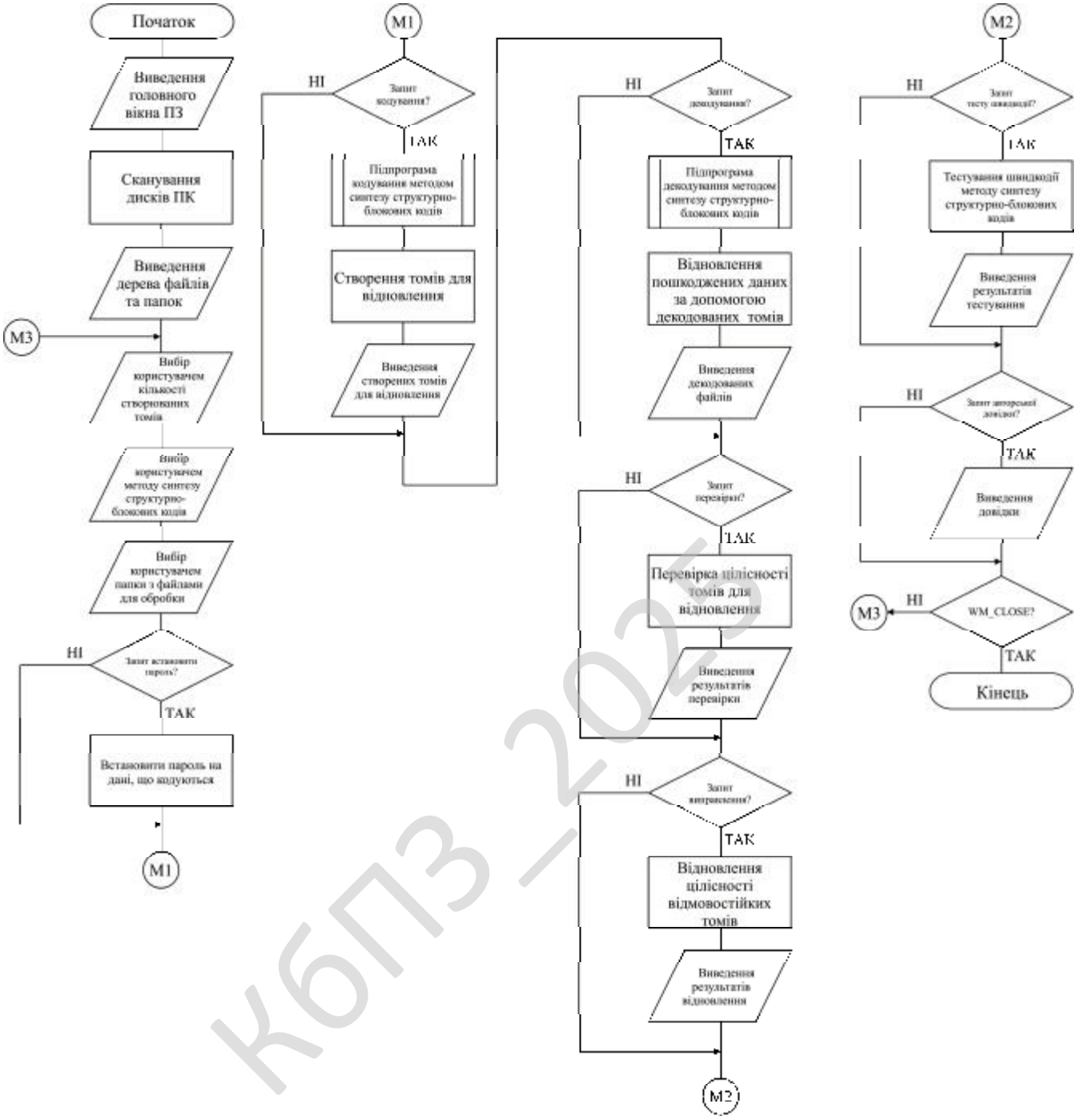

Рисунок 4.1 – Блок-схема основної програми

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.

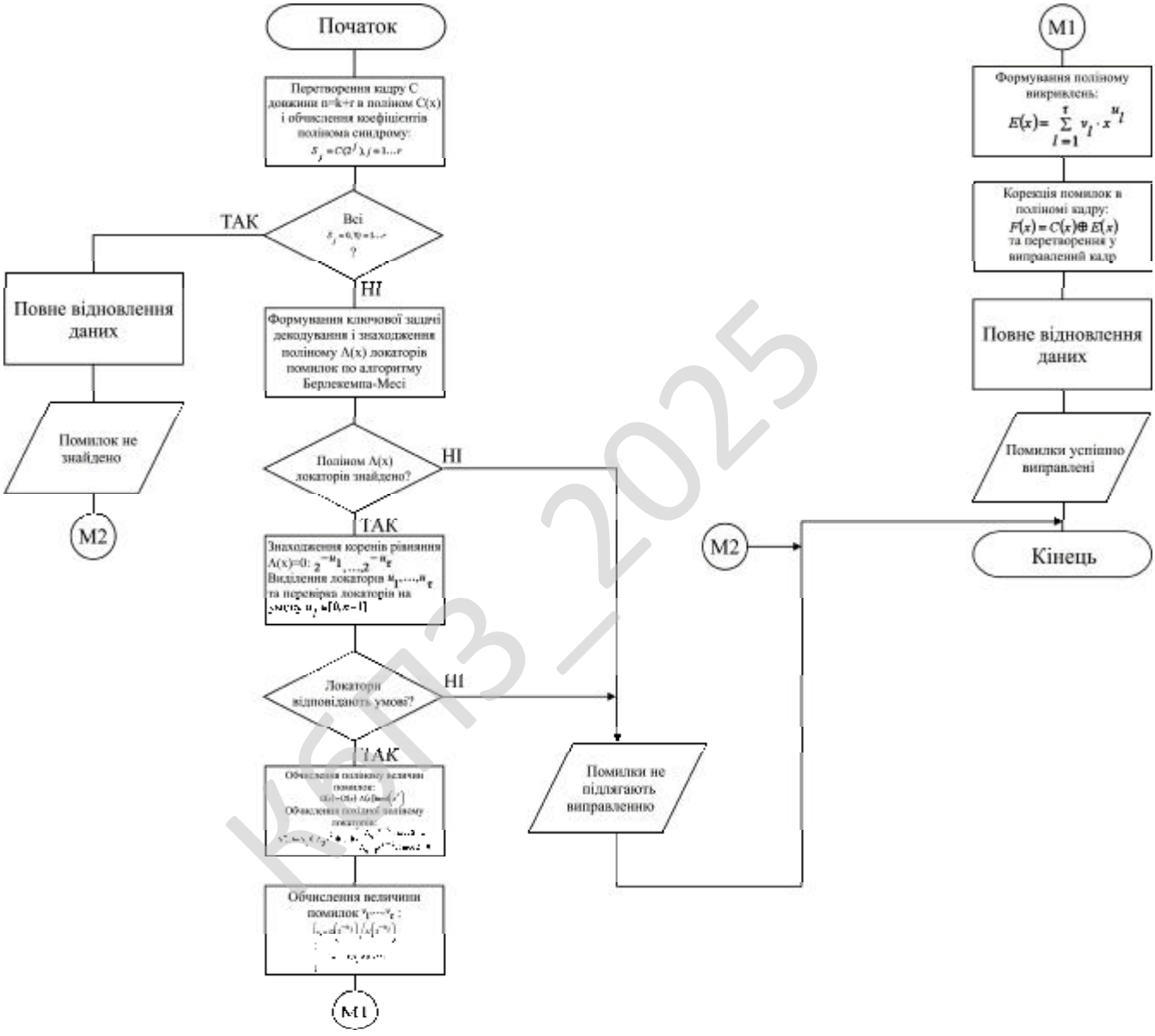

Рисунок 4.2 – Блок-схема роботи підпрограми

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 вихідного коду, яка виконує вищеперераховані дії.

```
/*--------------------------------
* синтез структурно-блокових кодів для
* спеціалізованих автоматизованих систем управління
* ========================
```

```
*
* кодуємі дані передаються через масив data[i], де i = 0...(k - 1),
* а згенеровані символи парності заносяться в масив b[0]...b[2 * t - 1].
* Вихідні й результуючі дані повинні бути представлені в поліноміальної
* формі (тобто у звичайній формі машинного подання даних).
* кодування виробляється з використанням сдвигового feedback-регістра,
* заповненого відповідними елементами масиву g[] з породженим
* поліномом усередині, процедура генерації якого вже обговорювалася в
* попередній главі.
* згенероване кодове слово описується наступною формулою:
* C(x) = data(x) * x ^ (n - k) + b(x), де ^ означає зведення в ступінь
*
-------------------------------------*/
encode_rs()
{
       int i, j;
       int feedback;
// ініціалізація поля біт парності нулями
       for (i = 0; i < n - k; i++) b[i] = 0;
// обробляємо всі символи
// вихідних даних праворуч ліворуч
       for (i = k - 1; i >= 0; i-i--)
       {
// готовимо (data[i] + b[n - k - 1]) до множення на g[i]
// тобто складаємо черговий "захоплений" символ вихідних
// даних з молодшим символом біт парності (відповідного
// "регістру" b2 t-1, і переводимо його в
// індексну форму, зберігаючи результат у регістрі feedback;
// як ми вже говорили, сума двох індексів є добуток  поліномів
              feedback = index_of[data[i] ^ b[n - k - 1]];
 // є ще символи для обробки?
              if (feedback != -1)
              {
 // здійснюємо зрушення ланцюга bx-регістрів
                     for (j = n - k - 1; j > 0; j-j--)
  // якщо поточний коефіцієнт g - це дійсний
 // (тобто ненульовий коефіцієнт, те
  // множимо feedback на відповідний g-коефіцієнт
  // і складаємо його з наступних елементів ланцюжка
if (g[j] != -1) b[j] = b[j - 1] ^ alpha_to[(g[j] + feedback) % n];
                     else
   // якщо поточний коефіцієнт g - це нульовий коефіцієнт,
```

```
    // виконуємо одне лише зрушення без множення, переміщаючи
    // символ з одного m-регістра в інший
                              b[j] = b[j - 1];
    // закільцьовуємо вихідний символ у крайній лівий b 0-регістр
                        b[0] = alpha_to[(g[0] + feedback) % n];
                }
                else
                {
   // розподіл завершений,
   // здійснюємо останнє зрушення регістра,
   // на виході регістра буде частка, що губиться,
   // а в самому регістрі – шуканий залишок
                        for (j = n - k - 1; j > 0; j-j--) b[j] = b[j - 1]; b[0] = 0;
                }
        }
}
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм ДСТУ 28147:2009, що є класичним алгоритмом симетричного шифрування на основі мережі Фейстеля (рисунок 4.4). Даний алгоритм шифрує інформацію блоками по 64 біта (такі алгоритми називаються "блоковими"). Зміст мережі Фейстеля полягає в тому, що блок шифруемої інформації розбивається на два або більше субблоків, частина яких обробляється за певним законом, після чого результат цієї обробки накладається (операцією побітового додавання за модулем 2) на необроблювані субблоки. Потім субблоки міняються місцями, після чого обробляються знову й т.д. певне для кожного алгоритму число раз – раундів.

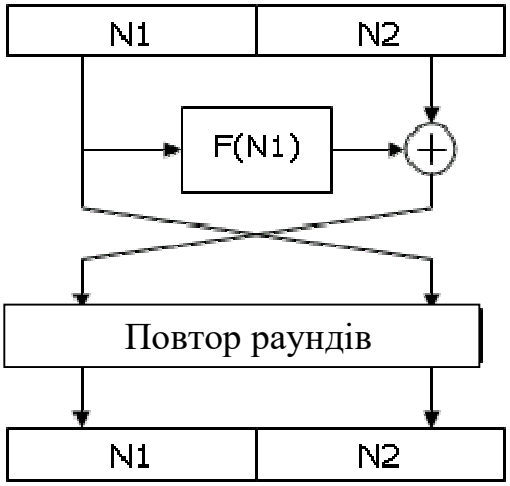

Рисунок 4.4 – Мережа Фейстеля

Основна відмінність алгоритмів симетричного шифрування друг від друга складається саме в різних функціях обробки субблоків. Дана функція часто називається "основним криптографічним перетворенням", оскільки саме вона несе основне навантаження при шифруванні інформації. Основне перетворення алгоритму ДСТУ 28147:2009 є досить простим, що забезпечує високу швидкодію алгоритму; у ньому виконуються наступні операції (рисунок 4.5).

Рисунок 4.5 – Основне перетворення алгоритму ДСТ 28147:2009

1. Додавання субблоку з певним фрагменом ключа шифрування за модулем 2^{32}. Кх – це 32-бітна частина ("підключ") 256-бітного ключа шифрування, якому можна представити як конкатенацію 8 підключів: К = $К_0К_1К_2К_3К_4К_5К_6$ $К_7$. Залежно від номера раунду й режиму роботи алгоритму (про їх – нижче), для даної операції вибирається один з підключів.

2. Таблична заміна. Для її виконання субблок розбивається на 8 4-бітних фрагментів, кожний з яких прогоняється через свою таблицю заміни. Таблиця заміни містить у певній послідовності значення від 0 до 15 (тобто всі варіанти значень 4-бітні фрагменти даних); на вхід таблиці подається блок даних, числове подання якого визначає номер вихідного значення. Наприклад, подається значення 5 на вхід наступної таблиці: "13 0 11 74 91 10 143 5 122 15 8 6". У результаті на виході виходить значення 9 (оскільки 0 заміняється на 13, 1 – на 0, 2 – на 11 і т.д.).

3. Побітове циклічне зрушення даних усередині субблока на 11 біт уліво.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

Розроблене програмне забезпечення синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління, складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Інструменти; Параметр; Довідка.

– Функції представлені у графічному вигляді (іконки).

– Вікна обрання групи даних.

– Навігаційного меню яке визивається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

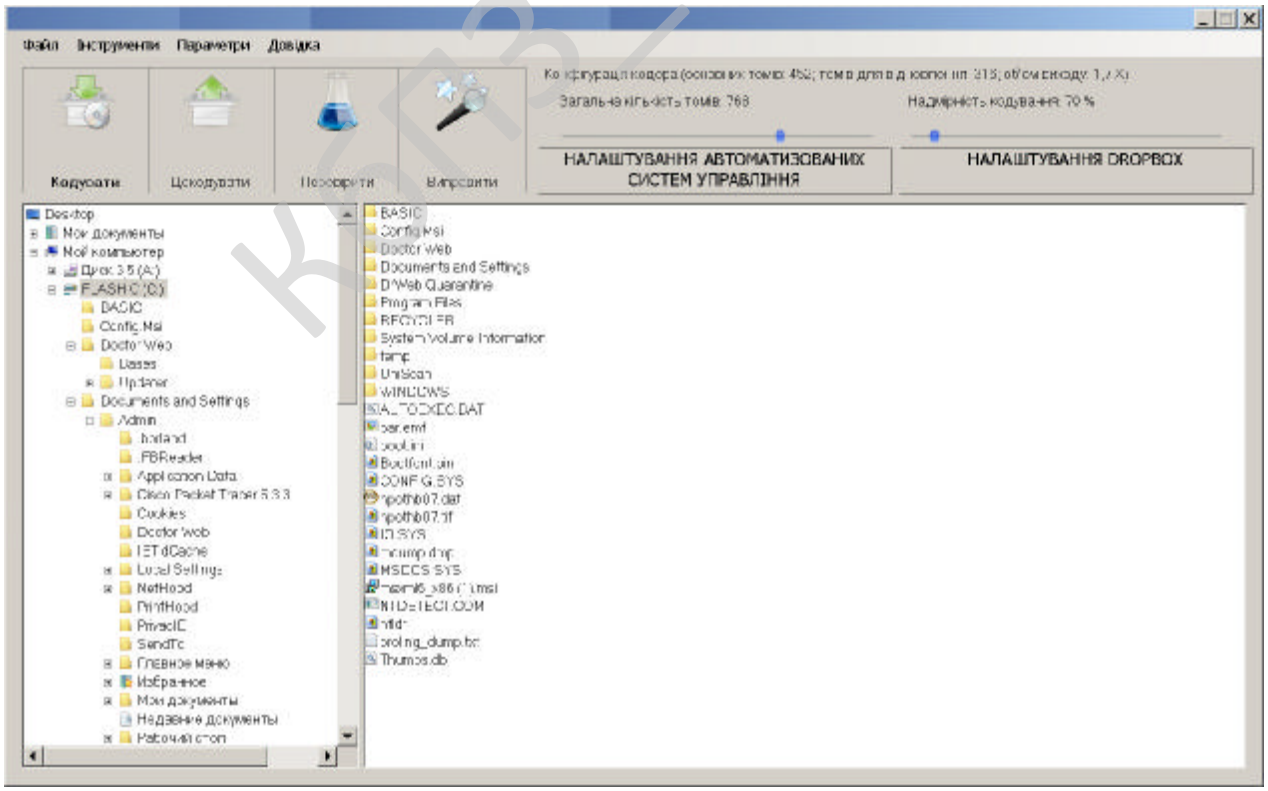

Рисунок 5.1 – Головне вікно розробленого ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

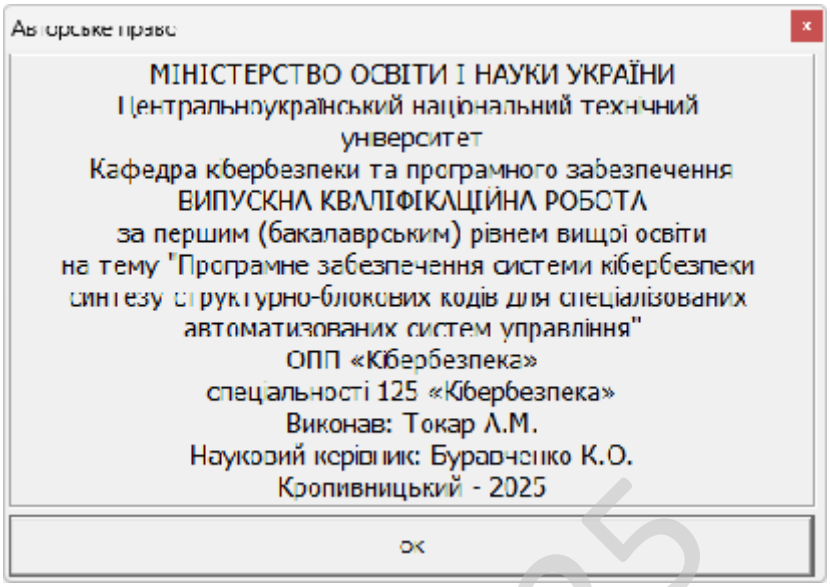

Рисунок 5.2 – Авторське право

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів. Проводилось тестування форматом білої скриньки та чорної скриньки.

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

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

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

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

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

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

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 ІТс).

– 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 ІТс;

– 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 – proprietary software.

Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію. Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт.

Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами. Визначенням власницького програмного забезпечення є не невідповідність хоча б одній з базових умов вільного програмного забезпечення. Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

– Досліджена система синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

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

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

Програма реалізована на мові високого рівня Visual C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 28147:2009.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

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

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

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

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

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

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

9. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

10. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

11. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

12. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnova, T., Prokopov, S., Bilanovych, A. «New Cost Function for S-boxes Generation by Simulated Annealing Algorithm». Lecture Notes on Data Engineering and Communications Technologies, 2023. vol 180. pp. 310-320. Springer, Cham.

13. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnov, O., Ulianovska, Y., Kobylianska, O. «Heuristic Search for Nonlinear Substitutions for Cryptographic Applications». *Lecture Notes on Data Engineering and Communications Technologies,* 2023. vol 180. Springer, Cham. pp. 288-298.

14. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

15. Kuznetsov O., Frontoni E., Kuznetsova Ye., Smirnov O., Chevardin V. «Achieving Enhanced Security in Biometric Authentication: A Rigorous Analysis of Code-Based Fuzzy Extractor». *CEUR Workshop Proceedings,* Volume 3624, 2023, pp. 330-339.

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

17. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

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.

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 57-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих 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, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Visual C#.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 63 аркуші.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 4.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Буравченко К.О.

Програмне забезпечення системи кібербезпеки синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем управління

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 61

Літера: РП

Кропивницький – 2025 року

Кодер структурно-блокових кодів

Файл RSRaidEncoder.cs - кодування алгоритмом структурно-блокових кодів

```
using System;
using System.Threading;

namespace RecoveryDisk
{
    /// <summary>
    /// Клас RAID-подібного кодера структурно-блокових кодів
    /// </summary>
    public class RSRaidEncoder : RSRaidBase
    {
        #region Construction & Destruction

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

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

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

        /// <summary>
        /// Розширений конструктор кодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="codecType">Тип кодека структурно-блокових кодів (по
типу матриці)</param>
        public RSRaidEncoder(int dataCount, int eccCount, int codecType)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, codecType);

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

        #endregion Construction & Destruction

        #region Public Operations

        /// <summary>
        /// Установка конфігурації кодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="codecType">Тип кодека структурно-блокових кодів (по
типу матриці)</param>
        /// <returns>Булевський прапор операції установки конфігурації</returns>
```

```
        public bool SetConfig(int dataCount, int eccCount, int codecType)
        {
            int maxVolCount;

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

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

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

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

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

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

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

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

                this.iterOfSecondStage = 0; // У 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.eRSType == (int)RSType.Dispersal)
                {
                    //...робимо формування дисперсної матриці "D"
                    if (!MakeDispersalMatrix())
                    {
                        // Указуємо, що кодер зконфігуровано некоректно
                        this.configIsOK = false;

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

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

                        return;
                    }
```

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

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

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

                        return;
                    }
                }

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

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

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

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

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

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

                    // Якщо зазначено, що потрібно вийти з потоку - виходимо
                    if (ManualResetEvent.WaitAll(this.exitEvent, 0, false))
                    {
                        // Указуємо, що кодер 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 (OnRSMatrixFormingFinish != null)
                {
                    //...повідомляємо, що екземпляр класу готовий до роботи
                    OnRSMatrixFormingFinish();
                }

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

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

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

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

        #endregion Private Operations
    }
}
```

Декодер структурно-блокових кодів

Файл RSRaidDecoder.cs - декодування алгоритмом структурно-блокових кодів

```
using System;
using System.Threading;

namespace RecoveryDisk
{
    /// <summary>
    /// Клас RAID-подібного декодера структурно-блокових кодів
    /// </summary>
    public class RSRaidDecoder : RSRaidBase
    {
        #region Data

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

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

        #endregion Data

        #region Construction & Destruction

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

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

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

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

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

```
        }

        #endregion Construction & Destruction

        #region Public Operations

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

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

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

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

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

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

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

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

```
                // залежить від типу використовуваної матриці
                if (this.eRSType == (int)RSType.Alternative)
                {
                    this.iterOfFirstStage = m;

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

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

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

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

                this.configIsOK = true;

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

            return this.configIsOK;
        }

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

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

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

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

                    data[i] = mulSum;

                } else
                {
```

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

            return true;
        }

        #endregion Public Operations

        #region Private Operations

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

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

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

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

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

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

                // Індекс розв'язного елемента
```

```
                int pivotIdx = k_n + k;

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

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

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

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

                    return false;
                }

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

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

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

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

                        int idx = i_n + j;

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

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

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

                    for (int j = 0; j < this.n; j++)
```

```
                    {
                        int idx = i_n + j;

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

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

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

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

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

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

                    return false;
                }
            }

            return true;
        }

        /// <summary>
        /// Обчислення логарифмів значень інвертованої матриці
        /// <summary>
        private void LogFCalc()
        {
            // Працюємо з рядками...
            for (int i = 0; i < this.n; i++)
            {
                // Зсув у масиві до елементів 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.eRSType == (int)RSType.Dispersal)
                {
                    //...робимо формування дисперсної матриці "D"
                    if (!MakeDispersalMatrix())
                    {
                        // Указуємо, що кодер зконфігуровано некоректно
                        this.configIsOK = false;

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

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

                        return;
                    }

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

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

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

                        return;
                    }
                }

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

            // Для кожного загубленого основного тому шукаємо том для
відновлення
            for (int i = 0, j = 0; i < dataVolMissCount; i++)
            {
```

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

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

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

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

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

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

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

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

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

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

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

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

                            this.FLog[i_n + i] = 1;

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

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

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

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

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

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

                        return;
                    }
                }

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

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

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

                    return;
                }

                // Обчислюємо логарифми елементів інвертованої матриці
                LogFCalc();

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

```
            }

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

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

        #endregion Private Operations

        #region Public Properties

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

                } else
                {
                    return null;
                }
            }
        }

        #endregion Public Properties
    }
}
```

Файл FileAnalyzer.cs - контроль цілісності набору файлів-томів

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

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

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

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

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

        #endregion Delegates

        #region Public Properties & Data

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

                } else
                {
                    return false;
                }
            }
        }

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

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

        } else
        {
            return false;
        }
    }
}

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

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

        } else
        {
            return false;
        }
    }
}

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

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

        } else
        {
            return null;
        }
    }
}

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

/// <summary>
/// Всі томи для відновлення коректні?
```

```
        /// </summary>
        public bool AllEccVolsOK
        {
            get
            {
                if (!InProcessing)
                {
                    return this.allEccVolsOK;

                } else
                {
                    return false;
                }
            }
        }

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

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

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

                            break;
                        }

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

                            break;
                        }

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

                            break;
                        }

                        case 3:
                        {
```

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

                            break;
                        }

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

                            break;
                        }
                    }

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

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

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

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

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

        #endregion Public Properties & Data

        #region Data

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

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

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

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

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

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

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

        /// <summary>
        /// Тип кодека структурно-блокових кодів (по типу використовуваної
матриці кодування)
        /// </summary>
        private int codecType;

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

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

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

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

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

        #endregion Data

        #region Construction & Destruction

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

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

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

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

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

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

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

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

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

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

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

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

        #endregion Construction & Destruction

        #region Public Operations

        /// <summary>
        /// Метод запуску потоку обробки обчислення й записи CRC64 у кінець
файлів
        /// </summary>
        /// <param name="path">Шлях до файлів для обробки</param>
        /// <param name="fileName">Ім'я файлу для обробки</param>
        /// <param name="dataCount">Конфігурація кількості основних
томів</param>
        /// <param name="eccCount">Конфігурація кількості томів для
відновлення</param>
        /// <param name="codecType">Тип кодека структурно-блокових кодів (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)RSConst.MaxVolCountAlt)
                )
            {
                // Активуємо індикатор актуального стану змінних-членів
                this.finished = true;

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

                return false;
            }

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

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

            // Зберігаємо тип кодека структурно-блокових кодів (по типу
використовуваної матриці кодування)
            this.codecType = codecType;

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

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

                // Повертаємо результат обробки
                return this.processedOK;
            }

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

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

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

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

            // Повідомляємо, що все нормально
            return true;
        }

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

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

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

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

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

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

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

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

```
                return false;
            }

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

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

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

                return false;
            }

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

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

            // Зберігаємо тип кодека структурно-блокових кодів (по типу
використовуваної матриці кодування)
            this.codecType = codecType;

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

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

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

                // Повертаємо результат обробки
                return this.processedOK;
            }

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

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

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

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

```
            // Повідомляємо, що все нормально
            return true;
        }

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

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

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

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

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

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

        #endregion Public Operations

        #region Private Operations

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

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

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

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

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

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

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

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

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

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

                            continue;
                        }

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

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

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

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

                            return;
                        }

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

```
                        if (eventIdx == 2)
                        {
                            //...exitимо із циклу очікування завершення (цього й
чекали в while(true)!)
                            break;
                        }

                    } // while(true)

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

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

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

                    return;
                }

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

                    } else
                    {
                        break;
                    }
                }

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

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

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

                    return;
                }

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

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

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

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

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

                    return;
                }
            }

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

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

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

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

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

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

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

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

            // Індекс у масиві томів
            int volListIdx = 0;

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

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

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

            // Ім'я файлу для обробки
            String fileName;

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

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

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

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

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

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

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

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

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

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

                                    continue;
                                }

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

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

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

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

                                    return;
                                }

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

                            } // while(true)

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

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

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

                            return;
                        }

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

                            } else
                            {
                                break;
```

```
                            }
                        }

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

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

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

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

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

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

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

                        return;
                    }
                }

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

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

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

            // Тепер, коли знаємо кількість ушкоджених основних томів,
            // потрібно просканувати всі файли для відновлення, і визначити
```

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

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

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

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

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

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

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

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

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

```
                                continue;
                            }

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

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

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

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

                                return;
                            }

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

                        } // while(true)

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

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

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

                        return;
                    }

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

                        } else
                        {
                            break;
                        }
                    }

                    // Указуємо, що том для відновлення коректний
                    if (this.eFileIntegrityCheck.ProcessedOK)
```

```
                        {
                            eccVolIsOK = true;
                        }

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

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

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

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

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

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

                        return;
                    }
                }

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

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

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

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

            // Виводимо статистику ушкоджень
```

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

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

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

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

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

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

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

                return;
            }

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

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

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

                return;
            }

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

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

```
                    while (altEccList[altEccListIdx] == -1)
                    {
                        altEccListIdx++;
                    }

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

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

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

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

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

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

        #endregion Private Operations
    }
}
```

Інтерфейс користувача

Файл MainForm.cs - головні вікно програми

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

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

        #region Windows Form Designer generated code

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

```
            System.Windows.Forms.TreeNode treeNode12 = new
System.Windows.Forms.TreeNode("WINDOWS", 18, 20, new
System.Windows.Forms.TreeNode[] {
            treeNode11});
            System.Windows.Forms.TreeNode treeNode13 = new
System.Windows.Forms.TreeNode("");
            System.Windows.Forms.TreeNode treeNode14 = new
System.Windows.Forms.TreeNode("WINDOWS.0", 18, 20, new
System.Windows.Forms.TreeNode[] {
            treeNode13});
            System.Windows.Forms.TreeNode treeNode15 = new
System.Windows.Forms.TreeNode("SYSTEM2 (C:)", 23, 24, new
System.Windows.Forms.TreeNode[] {
            treeNode2,
            treeNode4,
            treeNode6,
            treeNode8,
            treeNode10,
            treeNode12,
            treeNode14});
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(MainForm));
            this.menuStrip = new System.Windows.Forms.MenuStrip();
            this.fileToolStripMenuItem = new
System.Windows.Forms.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);
            //
            // quickextractionToolStripMenuItem
            //
            this.quickextractionToolStripMenuItem.Name =
"quickextractionToolStripMenuItem";
            this.quickextractionToolStripMenuItem.Size = new
System.Drawing.Size(216, 22);
            this.quickextractionToolStripMenuItem.Text = "Швидке вилучення
диску";
            this.quickextractionToolStripMenuItem.Click += new
System.EventHandler(this.quickextractionToolStripMenuItem_Click);
            //
            // helpToolStripMenuItem
            //
            this.helpToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.separatorToolStripMenuItem,
```

```
            this.aboutToolStripMenuItem});
            this.helpToolStripMenuItem.Name = "helpToolStripMenuItem";
            this.helpToolStripMenuItem.Size = new System.Drawing.Size(60, 17);
            this.helpToolStripMenuItem.Text = "Довідка";
            //
            // separatorToolStripMenuItem
            //
            this.separatorToolStripMenuItem.Name = "separatorToolStripMenuItem";
            this.separatorToolStripMenuItem.Size = new System.Drawing.Size(163,
6);
            //
            // aboutToolStripMenuItem
            //
            this.aboutToolStripMenuItem.Name = "aboutToolStripMenuItem";
            this.aboutToolStripMenuItem.Size = new System.Drawing.Size(166, 22);
            this.aboutToolStripMenuItem.Text = "Про програму...";
            this.aboutToolStripMenuItem.Click += new
System.EventHandler(this.aboutToolStripMenuItem_Click);
            //
            // coderConfigGroupBox
            //
            this.coderConfigGroupBox.Controls.Add(this.redundancyGroupBox);
            this.coderConfigGroupBox.Controls.Add(this.allVolCountGroupBox);
            this.coderConfigGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.coderConfigGroupBox.ForeColor =
System.Drawing.SystemColors.ControlText;
            this.coderConfigGroupBox.Location = new System.Drawing.Point(414,
26);
            this.coderConfigGroupBox.Name = "coderConfigGroupBox";
            this.coderConfigGroupBox.Size = new System.Drawing.Size(561, 98);
            this.coderConfigGroupBox.TabIndex = 5;
            this.coderConfigGroupBox.TabStop = false;
            this.coderConfigGroupBox.Text = "Конфігурація кодера (основних
томів: ...; томів для відновлення: ...; обсяг виход" +
                "в: ...)";
            //
            // redundancyGroupBox
            //
            this.redundancyGroupBox.Controls.Add(this.redundancyMacTrackBar);
            this.redundancyGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.redundancyGroupBox.ForeColor =
System.Drawing.SystemColors.ControlText;
            this.redundancyGroupBox.Location = new System.Drawing.Point(286,
21);
            this.redundancyGroupBox.Name = "redundancyGroupBox";
            this.redundancyGroupBox.Size = new System.Drawing.Size(264, 65);
            this.redundancyGroupBox.TabIndex = 4;
            this.redundancyGroupBox.TabStop = false;
            this.redundancyGroupBox.Text = "Надмірність кодування:";
            //
            // redundancyMacTrackBar
            //
            this.redundancyMacTrackBar.BackColor =
System.Drawing.Color.Transparent;
            this.redundancyMacTrackBar.BorderColor =
System.Drawing.SystemColors.ActiveBorder;
            this.redundancyMacTrackBar.Font = new System.Drawing.Font("Verdana",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.redundancyMacTrackBar.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(123)))), ((int)(((byte)(125)))),
((int)(((byte)(123)))));
            this.redundancyMacTrackBar.IndentHeight = 6;
            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.toolTip.SetToolTip(this.redundancyMacTrackBar, "Чим більше
надмірність кодування - том вище відмовостійкість, алі більше й обсяг " +
                    "отриманого набору томів");
            this.redundancyMacTrackBar.TrackerColor =
System.Drawing.Color.RoyalBlue;
            this.redundancyMacTrackBar.TrackerSize = new System.Drawing.Size(10,
16);
            this.redundancyMacTrackBar.TrackLineColor =
System.Drawing.Color.FromArgb(((int)(((byte)(90)))), ((int)(((byte)(93)))),
((int)(((byte)(90)))));
            this.redundancyMacTrackBar.TrackLineHeight = 3;
            this.redundancyMacTrackBar.Value = 19;
            this.redundancyMacTrackBar.ValueChanged += new
EConTech.Windows.MACUI.ValueChangedHandler(this.redundancyMacTrackBar_ValueChang
ed);
            this.redundancyMacTrackBar.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.redundancyMacTrackBar_MouseUp);
            //
            // allVolCountGroupBox
            //
            this.allVolCountGroupBox.Controls.Add(this.allVolCountMacTrackBar);
            this.allVolCountGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.allVolCountGroupBox.ForeColor =
System.Drawing.SystemColors.ControlText;
            this.allVolCountGroupBox.Location = new System.Drawing.Point(12,
21);
            this.allVolCountGroupBox.Name = "allVolCountGroupBox";
            this.allVolCountGroupBox.Size = new System.Drawing.Size(264, 65);
            this.allVolCountGroupBox.TabIndex = 3;
            this.allVolCountGroupBox.TabStop = false;
            this.allVolCountGroupBox.Text = "Загальна кількість томів:";
            //
            // allVolCountMacTrackBar
            //
            this.allVolCountMacTrackBar.BackColor =
System.Drawing.Color.Transparent;
            this.allVolCountMacTrackBar.BorderColor =
System.Drawing.SystemColors.ActiveBorder;
            this.allVolCountMacTrackBar.Font = new
System.Drawing.Font("Verdana", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.allVolCountMacTrackBar.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(123)))), ((int)(((byte)(125)))),
((int)(((byte)(123)))));
            this.allVolCountMacTrackBar.IndentHeight = 6;
            this.allVolCountMacTrackBar.Location = new System.Drawing.Point(6,
24);
            this.allVolCountMacTrackBar.Maximum = 24;
            this.allVolCountMacTrackBar.Minimum = 0;
            this.allVolCountMacTrackBar.Name = "allVolCountMacTrackBar";
            this.allVolCountMacTrackBar.Size = new System.Drawing.Size(252, 28);
            this.allVolCountMacTrackBar.TabIndex = 5;
            this.allVolCountMacTrackBar.TextTickStyle =
System.Windows.Forms.TickStyle.None;
            this.allVolCountMacTrackBar.TickColor =
System.Drawing.Color.FromArgb(((int)(((byte)(148)))), ((int)(((byte)(146)))),
((int)(((byte)(148)))));
```

```
            this.allVolCountMacTrackBar.TickHeight = 4;
            this.allVolCountMacTrackBar.TickStyle =
System.Windows.Forms.TickStyle.None;
            this.toolTip.SetToolTip(this.allVolCountMacTrackBar, "Чим більше
томів - том повільніше обробка й вище відмовостійкість");
            this.allVolCountMacTrackBar.TrackerColor =
System.Drawing.Color.RoyalBlue;
            this.allVolCountMacTrackBar.TrackerSize = new
System.Drawing.Size(10, 16);
            this.allVolCountMacTrackBar.TrackLineColor =
System.Drawing.Color.FromArgb(((int)(((byte)(90)))), ((int)(((byte)(93)))),
((int)(((byte)(90)))));
            this.allVolCountMacTrackBar.TrackLineHeight = 3;
            this.allVolCountMacTrackBar.Value = 14;
            this.allVolCountMacTrackBar.ValueChanged += new
EConTech.Windows.MACUI.ValueChangedHandler(this.allVolCountMacTrackBar_ValueChan
ged);
            this.allVolCountMacTrackBar.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.allVolCountMacTrackBar_MouseUp);
            //
            // toolTip
            //
            this.toolTip.AutomaticDelay = 2000;
            this.toolTip.AutoPopDelay = 20000;
            this.toolTip.InitialDelay = 2000;
            this.toolTip.ReshowDelay = 1000;
            //
            // browser
            //
            this.browser.ListViewMode = System.Windows.Forms.View.List;
            this.browser.Location = new System.Drawing.Point(12, 131);
            this.browser.Name = "browser";
            treeNode1.Name = "";
            treeNode1.Text = "";
            treeNode2.ImageIndex = 18;
            treeNode2.Name = "Documents and Settings";
            treeNode2.SelectedImageIndex = 20;
            treeNode2.Text = "Documents and Settings";
            treeNode3.Name = "";
            treeNode3.Text = "";
            treeNode4.ImageIndex = 18;
            treeNode4.Name = "Downloads";
            treeNode4.SelectedImageIndex = 20;
            treeNode4.Text = "Downloads";
            treeNode5.Name = "";
            treeNode5.Text = "";
            treeNode6.ImageIndex = 18;
            treeNode6.Name = "Program Files";
            treeNode6.SelectedImageIndex = 20;
            treeNode6.Text = "Program Files";
            treeNode7.Name = "";
            treeNode7.Text = "";
            treeNode8.ImageIndex = 18;
            treeNode8.Name = "RapidDriver";
            treeNode8.SelectedImageIndex = 20;
            treeNode8.Text = "RapidDriver";
            treeNode9.Name = "";
            treeNode9.Text = "";
            treeNode10.ImageIndex = 18;
            treeNode10.Name = "Server";
            treeNode10.SelectedImageIndex = 20;
            treeNode10.Text = "Server";
            treeNode11.Name = "";
            treeNode11.Text = "";
            treeNode12.ImageIndex = 18;
            treeNode12.Name = "WINDOWS";
            treeNode12.SelectedImageIndex = 20;
            treeNode12.Text = "WINDOWS";
            treeNode13.Name = "";
```

```
            treeNode13.Text = "";
            treeNode14.ImageIndex = 18;
            treeNode14.Name = "WINDOWS.0";
            treeNode14.SelectedImageIndex = 20;
            treeNode14.Text = "WINDOWS.0";
            treeNode15.ImageIndex = 23;
            treeNode15.Name = "SYSTEM2 (C:)";
            treeNode15.SelectedImageIndex = 24;
            treeNode15.Text = "SYSTEM2 (C:)";
            this.browser.SelectedNode = treeNode15;
            this.browser.ShowFoldersButton = false;
            this.browser.ShowNavigationBar = false;
            this.browser.Size = new System.Drawing.Size(962, 432);
            this.browser.SplitterDistance = 398;
            this.browser.StartUpDirectory = FileBrowser.SpecialFolders.Other;
            this.browser.StartUpDirectoryOther = "C:\\";
            this.browser.TabIndex = 0;
            //
            // repairButton
            //
            this.repairButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.repairButton.Image =
((System.Drawing.Image)(resources.GetObject("repairButton.Image")));
            this.repairButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.repairButton.Location = new System.Drawing.Point(308, 27);
            this.repairButton.Name = "repairButton";
            this.repairButton.Size = new System.Drawing.Size(100, 97);
            this.repairButton.TabIndex = 3;
            this.repairButton.Text = "Виправити";
            this.repairButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.toolTip.SetToolTip(this.repairButton, "Відновити цілісність
відмовостійкого набору томів");
            this.repairButton.UseVisualStyleBackColor = true;
            this.repairButton.Click += new
System.EventHandler(this.repairButton_Click);
            //
            // testButton
            //
            this.testButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.testButton.Image =
((System.Drawing.Image)(resources.GetObject("testButton.Image")));
            this.testButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.testButton.Location = new System.Drawing.Point(210, 27);
            this.testButton.Name = "testButton";
            this.testButton.Size = new System.Drawing.Size(100, 97);
            this.testButton.TabIndex = 4;
            this.testButton.Text = "Перевірити";
            this.testButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.toolTip.SetToolTip(this.testButton, "Перевірити на наявність
помилок відмовостійкий набір томів");
            this.testButton.UseVisualStyleBackColor = true;
            this.testButton.Click += new
System.EventHandler(this.testButton_Click);
            //
            // recoverButton
            //
            this.recoverButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.recoverButton.Image =
((System.Drawing.Image)(resources.GetObject("recoverButton.Image")));
            this.recoverButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.recoverButton.Location = new System.Drawing.Point(111, 27);
            this.recoverButton.Name = "recoverButton";
            this.recoverButton.Size = new System.Drawing.Size(100, 97);
            this.recoverButton.TabIndex = 2;
```

```
            this.recoverButton.Text = "Декодувати";
            this.recoverButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.toolTip.SetToolTip(this.recoverButton, "Розкодувати файли з
набору томів з корекцією помилок");
            this.recoverButton.UseVisualStyleBackColor = true;
            this.recoverButton.Click += new
System.EventHandler(this.recoverButton_Click);
            //
            // protectButton
            //
            this.protectButton.BackColor = System.Drawing.SystemColors.Control;
            this.protectButton.FlatStyle = System.Windows.Forms.FlatStyle.Popup;
            this.protectButton.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.protectButton.Image =
((System.Drawing.Image)(resources.GetObject("protectButton.Image")));
            this.protectButton.ImageAlign =
System.Drawing.ContentAlignment.TopCenter;
            this.protectButton.Location = new System.Drawing.Point(12, 27);
            this.protectButton.Name = "protectButton";
            this.protectButton.Size = new System.Drawing.Size(100, 97);
            this.protectButton.TabIndex = 1;
            this.protectButton.Text = "Кодувати";
            this.protectButton.TextAlign =
System.Drawing.ContentAlignment.BottomCenter;
            this.toolTip.SetToolTip(this.protectButton, "Закодувати файли у
відмовостійкому форматі в даній директорії");
            this.protectButton.UseVisualStyleBackColor = false;
            this.protectButton.Click += new
System.EventHandler(this.protectButton_Click);
            //
            // exitToolStripMenuItem
            //
            this.exitToolStripMenuItem.Image =
global::RecoveryDisk.Properties.Resources.Exit;
            this.exitToolStripMenuItem.Name = "виходToolStripMenuItem";
            this.exitToolStripMenuItem.Size = new System.Drawing.Size(112, 22);
            this.exitToolStripMenuItem.Text = "Вихід";
            this.exitToolStripMenuItem.Click += new
System.EventHandler(this.exitToolStripMenuItem_Click);
            //
            // testspeedToolStripMenuItem
            //
            this.testspeedToolStripMenuItem.Image =
global::RecoveryDisk.Properties.Resources.StartBenchmark;
            this.testspeedToolStripMenuItem.ImageScaling =
System.Windows.Forms.ToolStripItemImageScaling.None;
            this.testspeedToolStripMenuItem.Name =
"тестбистродействияToolStripMenuItem";
            this.testspeedToolStripMenuItem.Size = new System.Drawing.Size(161,
22);
            this.testspeedToolStripMenuItem.Text = "Тест швидкодії";
            this.testspeedToolStripMenuItem.Click += new
System.EventHandler(this.testspeedToolStripMenuItem_Click);
            //
            // MainForm
            //
            this.AutoScaleDimensions = new System.Drawing.Size(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(986, 576);
            this.Controls.Add(this.coderConfigGroupBox);
            this.Controls.Add(this.repairButton);
            this.Controls.Add(this.testButton);
            this.Controls.Add(this.recoverButton);
            this.Controls.Add(this.protectButton);
            this.Controls.Add(this.browser);
            this.Controls.Add(this.menuStrip);
```

```
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.MainMenuStrip = this.menuStrip;
            this.MaximizeBox = false;
            this.Name = "MainForm";
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Кодек структурно-блокових кодів - підвищення надійності
зберігання даних на CD/DVD";
            this.Load += new System.EventHandler(this.MainForm_Load);
            this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.MainForm_FormClosing);
            this.menuStrip.ResumeLayout(false);
            this.menuStrip.PerformLayout();
            this.coderConfigGroupBox.ResumeLayout(false);
            this.redundancyGroupBox.ResumeLayout(false);
            this.redundancyGroupBox.PerformLayout();
            this.allVolCountGroupBox.ResumeLayout(false);
            this.allVolCountGroupBox.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.MenuStrip menuStrip;
        private System.Windows.Forms.ToolStripMenuItem fileToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem helpToolStripMenuItem;
        private System.Windows.Forms.Button protectButton;
        private System.Windows.Forms.Button recoverButton;
        private System.Windows.Forms.ToolStripMenuItem exitToolStripMenuItem;
        private System.Windows.Forms.ToolStripSeparator
separatorToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem aboutToolStripMenuItem;
        private System.Windows.Forms.Button repairButton;
        private System.Windows.Forms.Button testButton;
        private System.Windows.Forms.GroupBox coderConfigGroupBox;
        private System.Windows.Forms.GroupBox redundancyGroupBox;
        private System.Windows.Forms.GroupBox allVolCountGroupBox;
        private EConTech.Windows.MACUI.MACTrackBar allVolCountMacTrackBar;
        private EConTech.Windows.MACUI.MACTrackBar redundancyMacTrackBar;
        private System.Windows.Forms.ToolTip toolTip;
        private System.Windows.Forms.ToolStripMenuItem
instrumentToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
testspeedToolStripMenuItem;
        internal FileBrowser.Browser browser;
        private System.Windows.Forms.ToolStripMenuItem
adjustmentToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
encodingfilterToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem
quickextractionToolStripMenuItem;
    }
}
```

Файл ProcessForm.cs - вікно кодування/декодування

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

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

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(ProcessForm));
            this.processPriorityGroupBox = new System.Windows.Forms.GroupBox();
            this.processPriorityComboBox = new System.Windows.Forms.ComboBox();
            this.processGroupBox = new System.Windows.Forms.GroupBox();
            this.processProgressBar = new System.Windows.Forms.ProgressBar();
            this.fileAnalyzeStatGroupBox = new System.Windows.Forms.GroupBox();
            this.percOfAltEccLabel = new System.Windows.Forms.Label();
            this.percOfDamageLabel = new System.Windows.Forms.Label();
            this.percOfAltEccLabel_ = new System.Windows.Forms.Label();
            this.percOfDamageLabel_ = new System.Windows.Forms.Label();
            this.logGroupBox = new System.Windows.Forms.GroupBox();
            this.logListBox = new System.Windows.Forms.ListBox();
            this.countGroupBox = new System.Windows.Forms.GroupBox();
            this.errorCountLabel = new System.Windows.Forms.Label();
            this.okCountLabel = new System.Windows.Forms.Label();
            this.errorPictureBox = new System.Windows.Forms.PictureBox();
            this.okPictureBox = new System.Windows.Forms.PictureBox();
            this.errorCountLabel_ = new System.Windows.Forms.Label();
            this.okCountLabel_ = new System.Windows.Forms.Label();
            this.toolTip = new System.Windows.Forms.ToolTip(this.components);
            this.stopButtonXP = new PinkieControls.ButtonXP();
            this.pauseButtonXP = new PinkieControls.ButtonXP();
            this.closingTimer = new System.Windows.Forms.Timer(this.components);
            this.processTimer = new System.Windows.Forms.Timer(this.components);
            this.processPriorityGroupBox.SuspendLayout();
            this.processGroupBox.SuspendLayout();
            this.fileAnalyzeStatGroupBox.SuspendLayout();
            this.logGroupBox.SuspendLayout();
            this.countGroupBox.SuspendLayout();

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

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

```
            this.SuspendLayout();
            //
            // processPriorityGroupBox
            //

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

```
            this.fileAnalyzeStatGroupBox.Controls.Add(this.percOfAltEccLabel_);
            this.fileAnalyzeStatGroupBox.Controls.Add(this.percOfDamageLabel_);
            this.fileAnalyzeStatGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.fileAnalyzeStatGroupBox.Location = new System.Drawing.Point(12,
216);
            this.fileAnalyzeStatGroupBox.Name = "fileAnalyzeStatGroupBox";
            this.fileAnalyzeStatGroupBox.Size = new System.Drawing.Size(459,
64);
            this.fileAnalyzeStatGroupBox.TabIndex = 0;
            this.fileAnalyzeStatGroupBox.TabStop = false;
            this.fileAnalyzeStatGroupBox.Text = "Результат аналізу цілісності
даних";
            //
            // percOfAltEccLabel
            //
            this.percOfAltEccLabel.AutoSize = true;
            this.percOfAltEccLabel.Location = new System.Drawing.Point(195, 41);
            this.percOfAltEccLabel.Name = "percOfAltEccLabel";
            this.percOfAltEccLabel.Size = new System.Drawing.Size(10, 13);
            this.percOfAltEccLabel.TabIndex = 0;
            this.percOfAltEccLabel.Text = "-";
            //
            // percOfDamageLabel
            //
            this.percOfDamageLabel.AutoSize = true;
            this.percOfDamageLabel.Location = new System.Drawing.Point(195, 20);
            this.percOfDamageLabel.Name = "percOfDamageLabel";
            this.percOfDamageLabel.Size = new System.Drawing.Size(10, 13);
            this.percOfDamageLabel.TabIndex = 0;
            this.percOfDamageLabel.Text = "-";
            //
            // percOfAltEccLabel_
            //
            this.percOfAltEccLabel_.AutoSize = true;
            this.percOfAltEccLabel_.Location = new System.Drawing.Point(7, 41);
            this.percOfAltEccLabel_.Name = "percOfAltEccLabel_";
            this.percOfAltEccLabel_.Size = new System.Drawing.Size(163, 13);
            this.percOfAltEccLabel_.TabIndex = 0;
            this.percOfAltEccLabel_.Text = "Резерв томів для відновлення:";
            //
            // percOfDamageLabel_
            //
            this.percOfDamageLabel_.AutoSize = true;
            this.percOfDamageLabel_.Location = new System.Drawing.Point(7, 20);
            this.percOfDamageLabel_.Name = "percOfDamageLabel_";
            this.percOfDamageLabel_.Size = new System.Drawing.Size(148, 13);
            this.percOfDamageLabel_.TabIndex = 0;
            this.percOfDamageLabel_.Text = "Всього пошкоджених томів:";
            //
            // logGroupBox
            //
            this.logGroupBox.Controls.Add(this.logListBox);
            this.logGroupBox.Location = new System.Drawing.Point(12, 80);
            this.logGroupBox.Name = "logGroupBox";
            this.logGroupBox.Size = new System.Drawing.Size(871, 130);
            this.logGroupBox.TabIndex = 0;
            this.logGroupBox.TabStop = false;
            this.logGroupBox.Text = "Лог процесу";
            //
            // logListBox
            //
            this.logListBox.BackColor = System.Drawing.SystemColors.Control;
            this.logListBox.BorderStyle = System.Windows.Forms.BorderStyle.None;
            this.logListBox.FormattingEnabled = true;
            this.logListBox.HorizontalScrollbar = true;
            this.logListBox.Location = new System.Drawing.Point(7, 23);
            this.logListBox.Name = "logListBox";
```

```
            this.logListBox.SelectionMode =
System.Windows.Forms.SelectionMode.None;
            this.logListBox.Size = new System.Drawing.Size(851, 91);
            this.logListBox.TabIndex = 0;
            this.logListBox.TabStop = false;
            this.logListBox.UseTabStops = false;
            this.logListBox.SelectedIndexChanged += new
System.EventHandler(this.logListBox_SelectedIndexChanged);
            //
            // countGroupBox
            //
            this.countGroupBox.Controls.Add(this.errorCountLabel);
            this.countGroupBox.Controls.Add(this.okCountLabel);
            this.countGroupBox.Controls.Add(this.errorPictureBox);
            this.countGroupBox.Controls.Add(this.okPictureBox);
            this.countGroupBox.Controls.Add(this.errorCountLabel_);
            this.countGroupBox.Controls.Add(this.okCountLabel_);
            this.countGroupBox.Location = new System.Drawing.Point(482, 216);
            this.countGroupBox.Name = "countGroupBox";
            this.countGroupBox.Size = new System.Drawing.Size(124, 64);
            this.countGroupBox.TabIndex = 0;
            this.countGroupBox.TabStop = false;
            this.countGroupBox.Text = "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 BenchmarkForm.cs - вікно тестування швидкодії алгоритму

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

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

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.coderConfigGroupBox = new System.Windows.Forms.GroupBox();
            this.eccCountLabel = new System.Windows.Forms.Label();
            this.dataCountLabel = new System.Windows.Forms.Label();
            this.eccCountLabel_ = new System.Windows.Forms.Label();
            this.dataCountLabel_ = new System.Windows.Forms.Label();
            this.coderSpeedGroupBox = new System.Windows.Forms.GroupBox();
            this.processedDataCountLabel = new System.Windows.Forms.Label();
            this.processedDataCountLabel_ = new System.Windows.Forms.Label();
            this.timeInTestLabel = new System.Windows.Forms.Label();
            this.timeInTestLabel_ = new System.Windows.Forms.Label();
            this.benchmarkTimer = new
System.Windows.Forms.Timer(this.components);
            this.toolTip = new System.Windows.Forms.ToolTip(this.components);
            this.pauseButtonXP = new PinkieControls.ButtonXP();
            this.closeButtonXP = new PinkieControls.ButtonXP();
            this.closingTimer = new System.Windows.Forms.Timer(this.components);
            this.coderConfigGroupBox.SuspendLayout();
            this.coderSpeedGroupBox.SuspendLayout();
            this.SuspendLayout();
            //
            // coderConfigGroupBox
            //
            this.coderConfigGroupBox.Controls.Add(this.eccCountLabel);
            this.coderConfigGroupBox.Controls.Add(this.dataCountLabel);
            this.coderConfigGroupBox.Controls.Add(this.eccCountLabel_);
            this.coderConfigGroupBox.Controls.Add(this.dataCountLabel_);
            this.coderConfigGroupBox.Location = new System.Drawing.Point(12,
90);
            this.coderConfigGroupBox.Name = "coderConfigGroupBox";
            this.coderConfigGroupBox.Size = new System.Drawing.Size(212, 72);
            this.coderConfigGroupBox.TabIndex = 0;
            this.coderConfigGroupBox.TabStop = false;
```

```
            this.coderConfigGroupBox.Text = "Конфігурація кодера";
            //
            // eccCountLabel
            //
            this.eccCountLabel.AutoSize = true;
            this.eccCountLabel.Location = new System.Drawing.Point(161, 47);
            this.eccCountLabel.Name = "eccCountLabel";
            this.eccCountLabel.Size = new System.Drawing.Size(37, 13);
            this.eccCountLabel.TabIndex = 0;
            this.eccCountLabel.Text = "65535";
            //
            // dataCountLabel
            //
            this.dataCountLabel.AutoSize = true;
            this.dataCountLabel.Location = new System.Drawing.Point(161, 25);
            this.dataCountLabel.Name = "dataCountLabel";
            this.dataCountLabel.Size = new System.Drawing.Size(37, 13);
            this.dataCountLabel.TabIndex = 0;
            this.dataCountLabel.Text = "65535";
            //
            // eccCountLabel_
            //
            this.eccCountLabel_.AutoSize = true;
            this.eccCountLabel_.Location = new System.Drawing.Point(11, 47);
            this.eccCountLabel_.Name = "eccCountLabel_";
            this.eccCountLabel_.Size = new System.Drawing.Size(125, 13);
            this.eccCountLabel_.TabIndex = 0;
            this.eccCountLabel_.Text = "Томів для відновлення:";
            //
            // dataCountLabel_
            //
            this.dataCountLabel_.AutoSize = true;
            this.dataCountLabel_.Location = new System.Drawing.Point(11, 25);
            this.dataCountLabel_.Name = "dataCountLabel_";
            this.dataCountLabel_.Size = new System.Drawing.Size(89, 13);
            this.dataCountLabel_.TabIndex = 0;
            this.dataCountLabel_.Text = "Основних томів:";
            //
            // coderSpeedGroupBox
            //
            this.coderSpeedGroupBox.Controls.Add(this.processedDataCountLabel);
            this.coderSpeedGroupBox.Controls.Add(this.processedDataCountLabel_);
            this.coderSpeedGroupBox.Controls.Add(this.timeInTestLabel);
            this.coderSpeedGroupBox.Controls.Add(this.timeInTestLabel_);
            this.coderSpeedGroupBox.Location = new System.Drawing.Point(12, 9);
            this.coderSpeedGroupBox.Name = "coderSpeedGroupBox";
            this.coderSpeedGroupBox.Size = new System.Drawing.Size(212, 72);
            this.coderSpeedGroupBox.TabIndex = 0;
            this.coderSpeedGroupBox.TabStop = false;
            this.coderSpeedGroupBox.Text = "Швидкість: - Мбайт/з";
            this.coderSpeedGroupBox.Enter += new
System.EventHandler(this.coderSpeedGroupBox_Enter);
            //
            // processedDataCountLabel
            //
            this.processedDataCountLabel.AutoSize = true;
            this.processedDataCountLabel.Location = new System.Drawing.Point(55,
47);
            this.processedDataCountLabel.Name = "processedDataCountLabel";
            this.processedDataCountLabel.Size = new System.Drawing.Size(10, 13);
            this.processedDataCountLabel.TabIndex = 0;
            this.processedDataCountLabel.Text = "-";
            //
            // processedDataCountLabel_
            //
            this.processedDataCountLabel_.AutoSize = true;
            this.processedDataCountLabel_.Location = new
System.Drawing.Point(11, 47);
            this.processedDataCountLabel_.Name = "processedDataCountLabel_";
```

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

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

        }

        #endregion

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

Файл About.cs - вікно довідки про програму

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

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

        #region Windows Form Designer generated code

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

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

```
            this.label1.Size = new System.Drawing.Size(144, 13);
            this.label1.TabIndex = 2;
            this.label1.Text = "БАКАЛАВРСЬКА РОБОТА";
            //
            // pictureBox1
            //
            this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(305, -28);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(133, 132);
            this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.Zoom;
            this.pictureBox1.TabIndex = 0;
            this.pictureBox1.TabStop = false;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(12, 45);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(49, 13);
            this.label2.TabIndex = 3;
            this.label2.Text = "на тему:";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(12, 79);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(257, 13);
            this.label3.TabIndex = 4;
            this.label3.Text = " Програмне забезпечення системи кібербезпеки
синтезу структурно-блокових кодів для спеціалізованих автоматизованих систем
управління ";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(12, 102);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(349, 13);
            this.label4.TabIndex = 5;
            this.label4.Text = "";
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(16, 140);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(139, 13);
            this.label5.TabIndex = 6;
            this.label5.Text = "Автор: студент Токар Артем Миколайович";
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Location = new System.Drawing.Point(54, 163);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(63, 13);
            this.label6.TabIndex = 7;
            this.label6.Text = "гр. КБ-21";
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Location = new System.Drawing.Point(16, 194);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(128, 13);
```

```
            this.label7.TabIndex = 8;
            this.label7.Text = "Керівник: Буравченко К.О.";
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Location = new System.Drawing.Point(16, 233);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(104, 13);
            this.label8.TabIndex = 9;
            this.label8.Text = "м. Кропивницький 2025";
            //
            // About
            //
            this.AutoScaleDimensions = new System.Drawing.Size(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(445, 258);
            this.Controls.Add(this.label8);
            this.Controls.Add(this.label7);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.pictureBox1);
            this.Name = "About";
            this.Text = "Про програму...";

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

        }

        #endregion

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