

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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
« ____ » _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи віртуалізації
додатків у ОС Linux”

Виконав здобувач вищої освіти
II курсу, групи КІ-23М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Серга А.В.
« ____ » _____ 2024 р.

Керівник проекту
доктор філософії (PhD)
_____ Дреєва Г.М.
« ____ » _____ 2024 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Рівень вищої освіти магістр
Галузь знань 12 "Інформаційні технології"
Спеціальність 123 "Комп'ютерна інженерія"
Освітньо-професійна (освітньо-наукова) програма "Комп'ютерна інженерія"

ЗАТВЕРДЖУЮ

Завідувач кафедри

д.т.н., проф.

Олексій СМІРНОВ

« 6 » вересня 2024 року

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

Сергію Андрію Володимировичу

(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux

2. Керівник роботи Дреєва Ганна Миколаївна, доктор філософії (PhD)

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

затверджені наказом вищого навчального закладу № 19-13 від 07.08.2024 року

3. Строк подання студентом роботи до захисту 2.12.2024 р.

4. Мета та завдання випускної кваліфікаційної роботи: Метою розробки є дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux

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

1. Призначення та область використання.

6. Наукова новизна.

2. Перегляд аналогічних існуючих систем.

7. Маркетингове та економічне обґрунтування IT-проєкту.

3. Опис і обґрунтування проєктних рішень.

8. Заходи з охорони праці та техніки безпеки.

4. Етапи програмування системи.

9. Висновки.

5. Впровадження системи в промислову експлуатацію

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

Наукова новизна

1 аркуш

Структурна схема системи

1 аркуш

Функціональна схема системи

1 аркуш

Діаграма процесів

1 аркуш

Блок-схема алгоритму роботи додатку

2 аркуша

Показники економічної ефективності

1 аркуш

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Доренська А.О.	05.10.2024	14.11.2024
Охорона праці	Марченко К.М., к.т.н., доцент	06.10.2024	16.11.2024

7. Дата видачі завдання « 6 » вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2024 р.	
3.	Розробка моделі компонента	20.10.2024 р.	
4.	Розробка структур даних	25.10.2024 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.10.2024 р.	
6.	Програмування алгоритмів	10.11.2024 р.	
7.	Розрахунок економічної ефективності	13.11.2024 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2024 р.	
9.	Оформлення ПЗ	17.11.2024 р.	
10.	Попередній захист роботи	2.12.2024 р.	

Дата видачі завдання
« 6 » вересня 2024 р.

Підпис керівника

_____ (прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2024 р.

Підпис здобувача

_____ (прізвище та ініціали)

АНОТАЦІЯ

Серга А.В. Дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віртуалізації додатків у ОС Linux.

Метою розробки є дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux.

Об'єктом дослідження є процес віртуалізації додатків у ОС Linux.

Предметом дослідження є методи віртуалізації додатків у ОС Linux.

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

Результат роботи – програмна реалізація системи віртуалізації додатків у ОС Linux.

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 Linux.

Програму розроблено в середовищі Visual C#.

Ключові слова: комп'ютерна інженерія, віртуалізація додатків, ОС Linux

ABSTRACT

Serga A.V. Research and software implementation of the application virtualization system in the OS Linux. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the system of virtualization of applications in the OS Linux.

The purpose of the development is the research and software implementation of the application virtualization system in the OS Linux.

The object of research is the process of virtualization of applications in the OS Linux.

The subject of research is methods of virtualization of applications in the OS Linux.

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

The result of the work is the software implementation of the application virtualization system in the OS Linux.

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

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

The program can be used on PCs with OS Linux.

The program was developed in the Visual C# environment.

Keywords: computer engineering, application virtualization, OS Linux

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	3
ВСТУП.....	4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	6
1.1 Призначення системи.....	6
1.2 Область застосування.....	7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	8
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.....	8
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування.....	10
2.3 Розгорнута постановка завдання	13
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	15
3.1 Опис функціонування системи	15
3.2 Розробка структурної схеми.....	20
3.3 Розробка функціональної схеми	23
3.4 Розробка діаграми процесів.....	35
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ.....	37
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	37
4.2 Захист розробленого програмного забезпечення.....	49
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	53
6 НАУКОВА НОВИЗНА	58

					ВКРМ-123.24.0039.00.00.ПЗ			
Вим	Арк.	№ докум.	Підп.	Дата	Дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux	Літ.	Аркуш	Аркушів
Розроб.	Серга А.В.					М	1	84
Перев.	Дресва Г.М.					ЦНТУ КІ-23М		
Н.контр.	Коваленко А.С.							
Затв.	Смірнов О.А.							

7	МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ІТ-ПРОЄКТУ	59
7.1	Визначення цільової аудиторії кінцевого готового продукту	59
7.2	Оцінка привабливості шляхом застосування методів експертних оцінок ...	60
7.3	Вибір методу оцінки вартості ПЗ	61
7.4	Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості.....	62
7.5	Пропозиція алгоритму просування проєкту розробки ПЗ	64
7.6	Оптимізація каналів збуту та шляхів реалізації ПЗ	64
7.7	Визначення ключових факторів успіху конкретного проєкту.....	67
8	ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ	68
8.1	Вступ.....	68
8.2	Шкідливі і небезпечні фактори при роботі з комп'ютером.....	69
8.3	Аналіз санітарно-гігієнічних умов праці на робочому місці програміста ...	70
8.4	Розробка заходів з умов поліпшення охорони праці	72
8.5	Розрахункова частина	73
9	ОСНОВНІ ВИСНОВКИ.....	76
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	78

КБПЗ-2024

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		2

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АВІ	– адміністратора віртуальної інфраструктури
АІБ	– адміністратор інформаційної безпеки
ВІ	– віртуальна інфраструктура
ЗЗІ	– засоби захисту інформації
ПД	– персональні дані
ПЗ	– програмне забезпечення
ЦЗОД	– центр зберігання й обробки даних
ЦОД	– центр обробки даних
DLP	– захист від витоків даних
IPS	– системи запобігання вторгнень
CIRC	– Cross Interleaved Reed Solomon Code
EAB	– Embedded Array Block, блок зосередженої пам'яті
ECC	– error-correcting code, код корекції помилок
FEC	– метод прямої корекції помилок
LDPC	– коди Галлагера
NAK	– негативне підтвердження

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

ВСТУП

Актуальність теми. Віртуальні інфраструктури вже довели свою перевагу при розгортанні серверних платформ, а при оснащенні робочих місць зручніше й вигідніше використовувати віртуальні робочі столи. Але це ще не кінець історії – тепер прийшов час віртуалізації додатків.

Актуальний ще п'ять років тому запит на покупку сервера для запуску окремого сервісу або бази даних виглядає сьогодні як мінімум безглуздо: усе більше ресурсів розміщається в хмарі або, принаймні, у віртуальному середовищі. Такий підхід дозволяє оптимально використовувати обчислювальні потужності і ємність зберігання даних. Втім, віртуальними стають уже цілі центри обробки даних – у портфелі рішень від лідерів ринку можна знайти засоби віртуалізації серверів, систем зберігання й мереж.

Інша тенденція, що стрімко трансформує ринок, – використання віртуальних робочих місць у рамках інфраструктури віртуальних робочих столів (VDI). Що впровадили VDI компанії можуть більше не піклуватися про те, які комп'ютери надати співробітникам: доступ до серверних ресурсів можливий навіть із тонких клієнтів. Однак і тут їсти де розгорнутися прихильникам подальшої оптимізації. Зокрема, багато елементів віртуальних машин дублюються – робочі столи використовують ту ж операційну систему й ті ж офісні додатки. І природно, у розроблювачів виникає спокуса об'єднати їх, щоб не зберігати множину схожих копій.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux.

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

- Огляд існуючих систем віртуалізації додатків у ОС Linux.
- Дослідження системи віртуалізації додатків у ОС Linux.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

– Програмна реалізація системи віртуалізації додатків у ОС Linux.

Об'єктом дослідження є процес віртуалізації додатків у ОС Linux.

Предметом дослідження є методи віртуалізації додатків у ОС Linux.

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

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

– Удосконалено метод віртуалізації додатків у ОС Linux.

– Розроблено вітчизняний продукт віртуалізації додатків у ОС Linux, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі віртуалізації додатків у ОС Linux.

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

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2024 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

У поточній ситуації наступним актуальним рівнем застосування віртуалізації є додатки. Адже якщо додатку однаково, у яких умовах працювати, його можна розгорнути на будь-якій віртуальній машині, на створюючи для цього окремих зліпків (знімків) системи. Великий крок у цьому напрямку був зроблений з появою технологій контейнерної віртуалізації, що компанія Parallels початку розвивати ще 15 років тому.

Контейнери являють собою ізольовані простори однієї й тої ж операційної системи, у кожному з яких виконується окремий додаток для конкретного користувача. Контейнер може бути організований як на віртуальній машині, так і на фізичному сервері, десктопі або мобільному пристрої. Більше того, в 2014 році – слідом за технологією контейнерної віртуалізації для Linux, історія успішних розробок якої до того моменту нараховувала понад 10 років, – вийшла версія Microsoft Windows Server із уже реалізованої, завдяки співробітництву з Docker, підтримкою контейнерів.

Контейнери, безперечно, перспективні з погляду безпеки й ефективності, оскільки додатки ізольовані, а завантаження ресурсів устаткування максимальні. Однак їхнє повсюдне впровадження в корпоративному середовищі, особливо коли мова йде про критично важливі навантаження, стримується через потенційні проблеми з відказостійкістю й безперервністю доступу. Що робити, якщо в тім або іншому контейнері відбувся збій? Потрібно чи заново завантажувати операційну систему й відновлювати дані або є спосіб простіше?

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

1.2 Область застосування

У співтоваристві програмістів давно обговорюється можливість надання користувачеві або процесу доступу до потрібного додатка або даних без прив'язки до конкретного контейнера. У цьому зв'язку виникає питання: як створити таку резервну копію додатка, щоб її можна було розгорнути в будь-якому іншому контейнері, на віртуальній машині або навіть на фізичній робочій станції? Оскільки додаток працює з певними областями пам'яті й звертається до системних процесів, для рішення цього завдання знадобиться дійсно комплексний інструмент, що працює на рівні як самих додатків, так і ядра ОС.

У цьому напрямку сьогодні тривають розробки різних технологій, що дозволяють забезпечити безперервне функціонування додатків і їхнє клонування без переривання роботи сервісів. У зв'язку з особливостями відкритих ОС поки такі можливості розглядаються тільки для операційних систем, заснованих на варіаціях ядра Linux. Що стосується продуктів Microsoft, подібних розробок для них поки не ведеться через закритість вихідного коду й не занадто активного прагнення Microsoft співробітничати з розроблювачами. Не виключено, що в майбутньому схожі проекти ми побачимо й в системі Windows, але поки мова про це не йде.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Більшість засобів, що дозволяють забезпечити перенос додатка з одного віртуального середовища в інше або його відновлення у випадку збоїти ОС, розробляються незалежними групами – як академічними, так і вхідними в комерційні структури, причому деякі з них орієнтуються тільки на певні модифікації ОС Linux. Що цікаво, практично всі вони використовують різні підходи до процесу так званої живої міграції. Більше того, для кожного проекту потрібно враховувати підтримувані технології контейнерної віртуалізації. Розглянемо докладніше найбільш успішні рішення даного завдання.

DMTSP

Один зі способів здійснення «живої міграції» і оперативного відновлення додатків пропонує команда розроблювачів з Північно-східного університету (США). Цей проект із відкритим кодом передбачає багатопоточне створення серій контрольних крапок для відновлення додатків. Distributed MultiThreaded CheckPointing (DMTSP) створює своє оточення в рамках Linux, де й стає можливою «живаюча міграція» підтримуваних додатків, у числі яких Python, Matlab, графічні інтерфейси робочих столів, MPI і багато хто інші. Проект активно розвивається, у його систему додаються нові додатки, але орієнтація на окремі додатки залишається його обмеженням. Втім, якщо потрібно забезпечити безперервну роботу й «живу міграцію» тільки графічних інтерфейсів робочих столів, цього цілком достатньо.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		8

BLCR

Спеціальна розробка Університету Берклі, Berkeley Lab Checkpoint/Restart (BLCR), використовує для своєї роботи модуль ядра. Завдяки цьому «живу міграцію» будь-яких додатків, що працюють під керуванням ОС, вдається забезпечити без створення окремої інфраструктури. Співробітники й студенти університету розробляють систему резервного копіювання й відновлення додатків і бібліотек MPI, а також інтерфейс керування. Однак вони зволіли самостійно доробити ядро, і в результаті BLCR виявився несумісний з багатьма версіями ядра Linux. Інакше кажучи, для того щоб скористатися можливостями BLCR, прийде завантажувати спеціальний дистрибутив.

PinPlay

Система PinPlay заснована на алгоритмах компанії Intel. Вона записує процесорні інструкції, робить знімки пам'яті, що використовувало той або інший додаток, а також зберігає необхідні йому бібліотеки. Потенціал даного рішення дозволяє оптимізувати роботу додатків у віртуальних середовищах, але, на жаль, здійснити повний перезапуск додатка не можна, тому що в PinPlay відсутні запис початкового стану додатка й інформація про використовувані об'єкти ядра. Проте за допомогою PinPlay можна прискорити відновлення після збою. Цілком імовірно, що після розширення функціонала PinPlay цей інструмент теж стане повноцінним рішенням для віртуалізації й «живої міграції» додатків.

CryoPid

Більше цікаве рішення запропоноване розроблювачами CryoPid, які спробували «поставити на паузу» будь-які додатки Linux, причому в користувальницькому просторі, тобто незалежно від ядра ОС. Однак, зштовхнувшись із обмеженими можливостями ядра й розходженнями процедур у різних версіях Linux, вони, на жаль, згорнули проект приблизно чотири роки тому. Дуже жаль.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		9

– 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 пряий доступ до пам'яті має вкрай важливе значення.

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

Архітектура платформи .NET Framework

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

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

При виконанні програми на Visual C# складання завантажується в середовище CLR залежно від відомостей у маніфесті. Далі, якщо вимоги безпеки

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

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 керування рядками для розбивки XML, і закінчуючи елементами керування Windows Forms. У звичайному додатку мовою Visual C# бібліотека класів .NET Framework інтенсивно використовується для "устрою" коду.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи віртуалізації додатків у ОС Linux.

В процесі розробки випускної кваліфікаційної роботи за другим

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		13

(магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		14

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У цьому розділі наведемо опис технологій, які дозволяють реалізувати віртуалізацію додатків у ОС Linux.

Переваги технологій «живої міграції» полягають у тому, що переносити можна практично що завгодно й куди завгодно. Як відзначають розроблювачі, є тільки дві вимоги. По-перше, можливості процесора на цільовій машині повинні бути не гірше, ніж на джерелі. Наприклад, якщо на джерелі підтримувалася технологія SSE, воно ж повинна бути й на цільовій системі, куди мігрує додаток. По-друге, ядро ОС на приймачі повинне бути «не гірше», чим на джерелі. Воно не обов'язково повинне бути тієї ж або більше пізньої версії, але в ньому повинна бути реалізована підтримка того ж інструмента «живої міграції», що ви використовуєте, а також бути вся необхідна для додатка функціональність, наприклад мережні протоколи або файлові системи.

Всі перераховані вище проекти вже застосовуються на практиці. Так, DMSTP використовується в Північно-східному університеті США для роботи з Matlab і Python, а також з іншими інструментами. BLCR – невід'ємна частина модифікації Linux, розповсюджуваної й використовуваної в Берклі, тому робочі навантаження дослідницьких підрозділів уже захищені на рівні додатків. CRIU, у свою чергу, крім інтеграції в дистрибутив RHEL 7, використовується в різних проектах, у тому числі в Tonic. Для даного інструмента навіть розроблена спеціальна надбудова r.haul, що дозволяє автоматизувати процес міграції, наприклад, у ЦОД. Однак всі ці рішення ще тільки розвиваються й перебувають у різному ступені зрілості.

З огляду на, що як мінімум три великі команди працюють над розвитком рішень «живої міграції» у віртуальних середовищах Linux, сьогодні можна

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		15

говорити про відносну незалежність додатків від конкретної системи й навіть від контейнера. Поки ці функції тільки починають застосовуватися комерційними компаніями, але, швидше за все, уже в найближчі роки ми побачимо підвищення попиту на дані технології за рахунок популяризації засобів контейнерної віртуалізації й росту вимог до забезпечення безперервності працюючих у них ІТ-процесів. А слідом за успіхами співтовариства «по живій міграції» OpenSource свої рішення запропонує й Microsoft, але до цього пройде чимало часу.

CRIU

CRIU (від англ. Checkpoint/Restore In Userspace) – програмне забезпечення для операційної системи Linux, що дозволяє створити ззовні під час виконання довільної програми контрольної крапки з можливістю поновлення роботи програми із цієї крапки, у тому числі, на іншому екземплярі операційної системи (функція живої міграції).

Головною особливістю проекту є його переважна реалізація в просторі користувальницьких процесів, а не в ядрі операційної системи (як, приміром, у проекті OpenVZ), а також можливість роботи з довільними процесами без додаткової підтримки з їх боку.

Можливості

Оскільки проект розробляється в рамках проекту OpenVZ, основною метою, переслідуваної розроблювачами, є підтримка міграції контейнерів. Проте, за допомогою CRIU істи можливість зберігати й відновлювати стан окремих процесів або їхніх груп.

За станом на середину 2010-х років підтримується дві архітектури – x 86-64 і ARM – і наступні об'єкти Linux:

- процеси, у тому числі їхня ієрархія, PID, ідентифікатори користувача й групи (uid, euid, sid, ...), системні права, потоки, стани;
- пам'ять додатків, у тому числі відображені файли, поділювані ділянки;
- відкриті файли;
- конвеєри, включаючи FIFO,

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

- сокети: сокети домену, TCP-сокети (у тому числі й у стані ESTABLISHED), UDP-сокети;
- System V IPC;
- таймери;
- сигнали;
- термінали;
- специфічні для Linux файлові дескриптори: signalfd, inotify, eventfd, eventpoll.

Вся необхідна підтримка ядра включена в ядро Linux версії 3.11.

Міграція TCP-з'єднань

Однієї з поставлених цілей проекту є можливість зберігати й відновлювати стан TCP-з'єднання, причому основний інтерес представляє випадок, коли процедурі збереження з наступним відновленням піддається тільки одна сторона з'єднання. Така постановка завдання виникла з основного сценарію використання CRIU компанією Parallels, при якому контейнер мігрує з однієї машини на іншу, і при цьому всі зовнішні з'єднання «переїжджають» разом з контейнером. Для досягнення поставленої мети був розроблений і впроваджений у ядро Linux версії 3.5 механізм, що одержав назву TCP repair mode^[8], у рамках якого реалізований набір операцій над TCP-сокетом, що дозволяють «розібрати» і «зібрати» його не запускаючи ніяких описаних у стандарті процедур по мережному обміні пакетами.

OpenVZ

Віртуалізація OpenVZ – це серверна технологія на рівні операційної системи, яка фактично базується на ядрі та ОС Linux. Віртуалізація на рівні ОС означає, що багато основних компонентів існують один раз на машині та використовуються всіма гостьовими системами (наприклад, ідентичне ядро). Це означає, що кожен контейнер використовує ту саму базову ОС, але все ще працює як безпечний ізольований контейнер Linux. Кожен контейнер працює та виконується так само, як окремий сервер; Контейнер можна перезавантажити

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		17

Комплект користувальницьких утиліт

Основними утилітами є:

– `vzctl` – Основна утиліта для керування контейнерами. Застосовується для створення, видалення, запуску, зупинки й перезапуску контейнерів, а також для застосування нових параметрів. До складу пакета також входять `vzlist`, `vzmigrate`, `vzcalc`, `vzcfgvalidate`, `vzmemcheck`, `vzcpucheck`, `vzpid`, `vzsplit` і деякі інші, менш значимі утиліти.

– `vzlist` – Друкує лістинг контейнерів. Підтримує вибірки по різних параметрах контейнерів і їхніх комбінацій.

– `vzmigrate` – Утиліта для `offline` і `online` міграції.

– `vzcfgvalidate` – Перевіряє вірність конфігураційного файлу контейнера.

– `vzmemcheck`, `vzcpucheck`, `vzcalc` – Служать для одержання інформації про використовувані ресурси.

– `vzsplit` – Служить для генерації конфігураційних файлів контейнерів. Дозволяє «розділити» фізичний сервер на зазначене число частин.

– `vzpid` – Дозволяє визначити, якому контейнеру належить процес. Як параметр приймає `pid`-номер процесу.

– `vzquota` – Утиліта для роботи з дисковою квотою контейнера.

– `vzubc` – Утиліта для заміни `cat /proc/user_beancounters`.

Також існує ряд утиліт або не підтримуваних офіційно, або «занедбаних»:

– `vzdump` – Неофіційна утиліта для створення `backup`'ов контейнерів.

Підтримує «гаряче» створення копій.

– `vzrpm`, `vz yum`, `vzpkg` – утиліти для роботи з `HardwareNode` з пакетною системою гостьового оточення й темплейтами.

– `vzpkg2` – Форк `vzpkg`, підтримує менеджмент `RedHat-based` і `Debian-based` гостьових оточень (`CentOS`, `Fedora`, `Debian`, `Ubuntu`). Як видно, проект занедбаний.

Об'єкти IPC

Поділювана пам'ять, семафори, повідомлення.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		19

3.2 Розробка структурної схеми

У даній роботі реалізовано програмне забезпечення віртуалізації додатків у ОС Linux у віртуалізованому хмарному центрі обробки даних з реалізацією віртуалізації додатків у ОС Linux.

Безпека віртуалізованого хмарного центру обробки даних з реалізацією віртуалізації додатків у ОС Linux в умовах збільшення навантаження

Оскільки всі сервера в віртуалізованому хмарному центрі обробки даних з реалізацією віртуалізації додатків у ОС Linux можна розділити на організаційні категорії відповідно до вимог замовників, то в ідеальному випадку, кожна логічна група хостів повинна бути безпечно обмежена від інших. Доступ між цими групами повинен бути або обмежений, або закритий відповідно до політики безпеки організації. У реальному житті це неможливо, оскільки межмережний екран повинен пропустити величезну кількість підключень до кожного сервера, обробляючи при цьому кожну сесію окремо. Саме фактор кількості нових сесій у секунду є визначальним на етапі вибору межмережного екрана для впровадження в віртуалізованому хмарному центрі обробки даних з реалізацією віртуалізації додатків у ОС Linux, і відстежити їхнє число дуже складно. Найчастіше можна судити лише про середнє значення швидкості нових підключень, а несподівані перегони можуть часово блокувати роботу пристрою або навіть привести до його відмови. Тому будь-який фаєрвол, впроваджуваний у віртуалізованому хмарному центрі обробки даних з реалізацією віртуалізації додатків у ОС Linux, повинен ефективно обробляти як постійне значення швидкості встановлення нових підключень, так і перегони навантаження, у кілька разів перевищуючі середні значення.

Після обробки підключення встановлюється з'єднання й кількість сесій обслуговуються пристроєм у конкретний момент визначає ще один важливий параметр при впровадженні пристрою безпеки – кількість установлених сесій.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		20

Велика кількість серверів у віртуалізованому хмарному центрі обробки даних з реалізацією віртуалізації додатків у ОС Linux при великій кількості підключених користувачів у результаті дає величезне число встановлених сесій. Тут можна відзначити дві важливі особливості – здатність пристрою ефективно обслуговувати вже встановлені сесії, а також, можливість обробляти нові. Неможливість створення нових з'єднань при обслуговуванні вже встановлених приведе до неприступності сервісів для деяких користувачів.

Після установки сесій ще одним фактором, що впливає на успішне проходження трафіку через фаєрвол, є загальна кількість пакетів у секунду, а також, загальна пропускна здатність для всіх сесій. Величина, характеризуюча кількість пакетів у секунду, важлива, у тому випадку, коли пристрій демонструє заявлену пропускну здатність тільки при обробці невеликої кількості пакетів великий довгі. Наприклад, число пакетів в 64-байтному потоці в 23 рази перевищує те ж число в 1500-байтному. Зміст полягає в тім, що фаєрвол може забезпечити задану пропускну здатність не для будь-якого типу трафіку. При проектуванні віртуалізованого хмарного центру обробки даних з реалізацією віртуалізації додатків у ОС Linux завжди потрібно враховувати реальні показники продуктивності фаєрволу.

Безпека в віртуалізованому хмарному центрі обробки даних з реалізацією віртуалізації додатків у ОС Linux вкрай важлива, тому що є ядром обробки інформації для більшості компаній. При її впровадженні дуже важливе логічне угруповання схожих сервісів, для взаємодії яких немає необхідності впроваджувати зайві міри безпеки. Наприклад, оптимально групувати веб-сервера, сервера додатків і сервера баз даних в одному логічному сегменті. Першим кроком при забезпеченні безпеки для серверів є обмеження можливих зв'язків до необхідного мінімуму, що забезпечить захист від неавторизованого доступу. Другим – впровадження технології запобігання вторгнень (IPS), що забезпечить перевірку мережного трафіку на наявність атак. Це особливо актуально в віртуалізованих хмарних центрах обробки даних з реалізацією віртуалізації додатків у ОС Linux, де уразливість сервісу може привести до його

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		21

неприступності або втрати важливої інформації. Через небезпеку таких наслідків впровадження IPS критично важливо, але є й деякі особливості – обмеження по місцю впровадження й продуктивності. Через обмежену пропускну здатність пристрій не завжди може обробити необхідна кількість трафіку при впровадженні в певному сегменті мережі, наприклад, ядрі віртуалізованого хмарного центру обробки даних з реалізацією віртуалізації додатків у ОС Linux, а також, неможливо вказати який трафік обробляти, а який немає. Тому, пріоритетної є селективна обробка трафіку й для визначення рівня необхідної продуктивності.

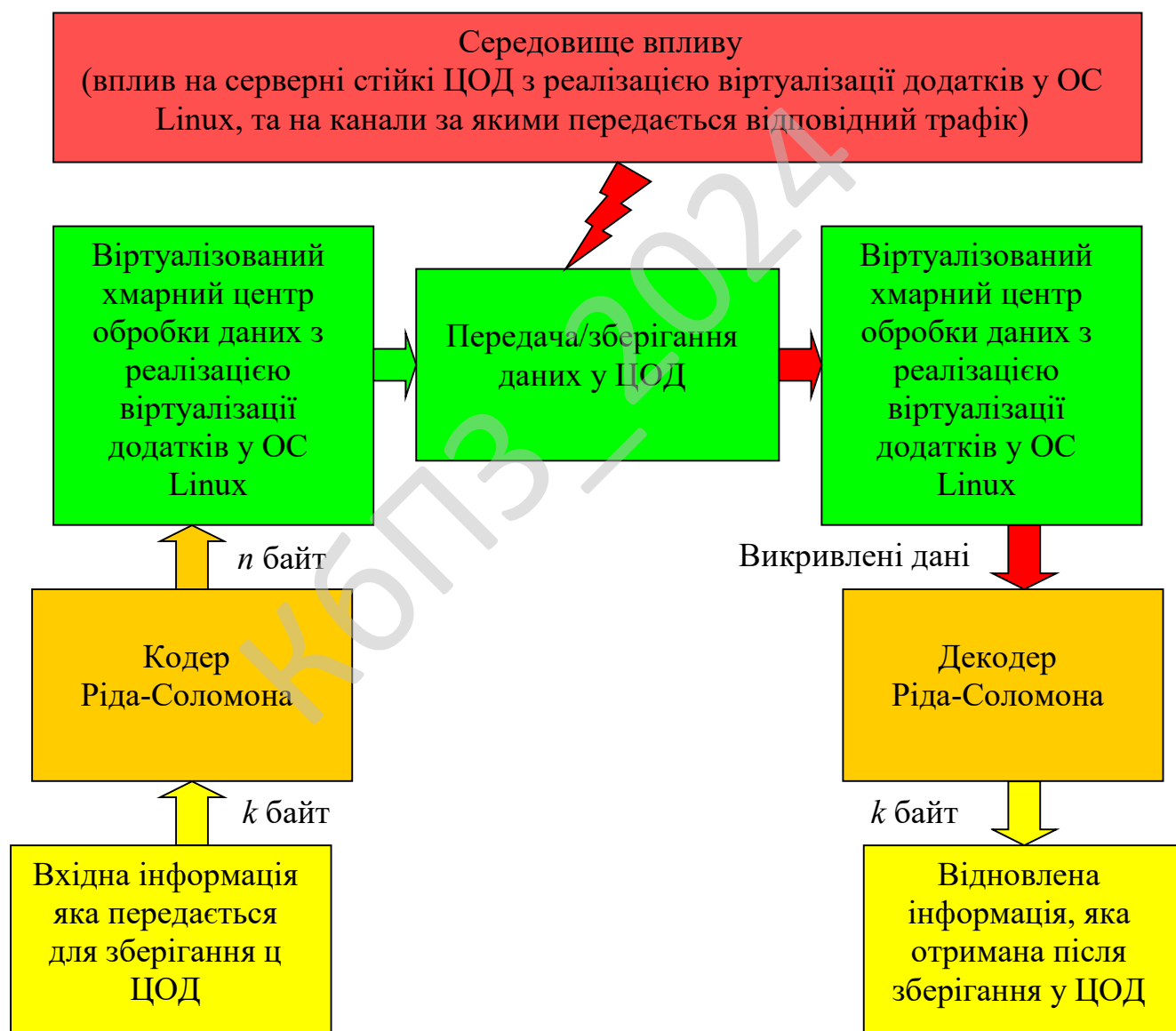


Рисунок 3.1 – Структурна схема системи

У сучасних віртуалізованих хмарних центрах обробки даних з реалізацією віртуалізації додатків у ОС Linux для повної з'язуємості мереж обов'язковим є використання протоколів динамічної маршрутизації. Це ж стосується й пристроїв безпеки, для яких цей функціонал не є першочерговим. Більше вузька підтримка протоколів динамічної маршрутизації може створити труднощі при впровадженні фаєрволу в інфраструктуру мережі віртуалізованого хмарного центру обробки даних з реалізацією віртуалізації додатків у ОС Linux.

Для реалізації перешкодостійкого зберігання інформації у мережних віртуалізованих хмарних центрах обробки даних з реалізацією віртуалізації додатків у ОС Linux, застосуємо алгоритм Ріда-Соломона.

3.3 Розробка функціональної схеми

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

У цей час широко використовується в системах відновлення даних на різних носіях, у тому числі й у віртуалізованих хмарних центрах обробки даних з реалізацією віртуалізації додатків у ОС Linux, при створенні архівів з інформацією для відновлення у випадку ушкоджень, у завадостійкому кодуванні.

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

Коди Ріда-Соломона є важливим частковим випадком БЧХ-коду, корінь полінома, що породжує, якого лежать у тім же полі, над яким і будується код

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

потрібно виконати операцію декодування й уже потім можна перевірити дані на зміст помилок. Таке кодування вимагає більші витрати ресурсів тільки на добування інформаційних даних, при цьому вони можуть бути без помилок.

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

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

Кодування

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

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

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

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

Результатом операції є результат додавання, перетворена обернено в

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		27

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

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

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

Коректність цього алгоритму випливає з наступних двох тверджень:

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

Знаходження кореня

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

Визначення характеру помилки і її виправлення

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

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

Структурна схема системи підвищення надійності зберігання даних у віртуалізованому хмарному центрі обробки даних з реалізацією віртуалізації додатків у ОС Linux за допомогою кодека Ріда–Соломона зображена на рисунку

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		30

3.2.

З цієї схеми ми бачимо, що над вхідними даними, перед записом у віртуалізованому хмарному центрі обробки даних з реалізацією віртуалізації додатків у ОС Linux, відбуваються перетворення кодеком Ріда-Соломона. Після кодування дані записуються у віртуалізований хмарний центр обробки даних з реалізацією віртуалізації додатків у ОС Linux. У якості носія окрім віртуалізованого хмарного центру обробки даних з реалізацією віртуалізації додатків у ОС Linux може використовуватися любий інший носій інформації. Після цього інформація зберігається на носієві.

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

На рисунку 3.2 зображена функціональна схема системи. З неї бачимо, що розроблена система підвищення стійкості до уражень інформації яка записана на дисках складається з наступних основних функціональних блоків:

- носій інформації – місце на сервері віртуалізованого хмарного центру обробки даних з реалізацією віртуалізації додатків у ОС Linux;
- кодер Ріда-Соломона, який використовується, коли відбувається запис інформації на носій інформації;
- декодер Ріда-Соломона, який використовується при читанні даних с відповідного носія інформації.

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

- дані, які потрібно зберегти у віртуалізованому хмарному центрі обробки даних з реалізацією віртуалізації додатків у ОС Linux;
- поліном для кодування;
- відмовостійки коди.

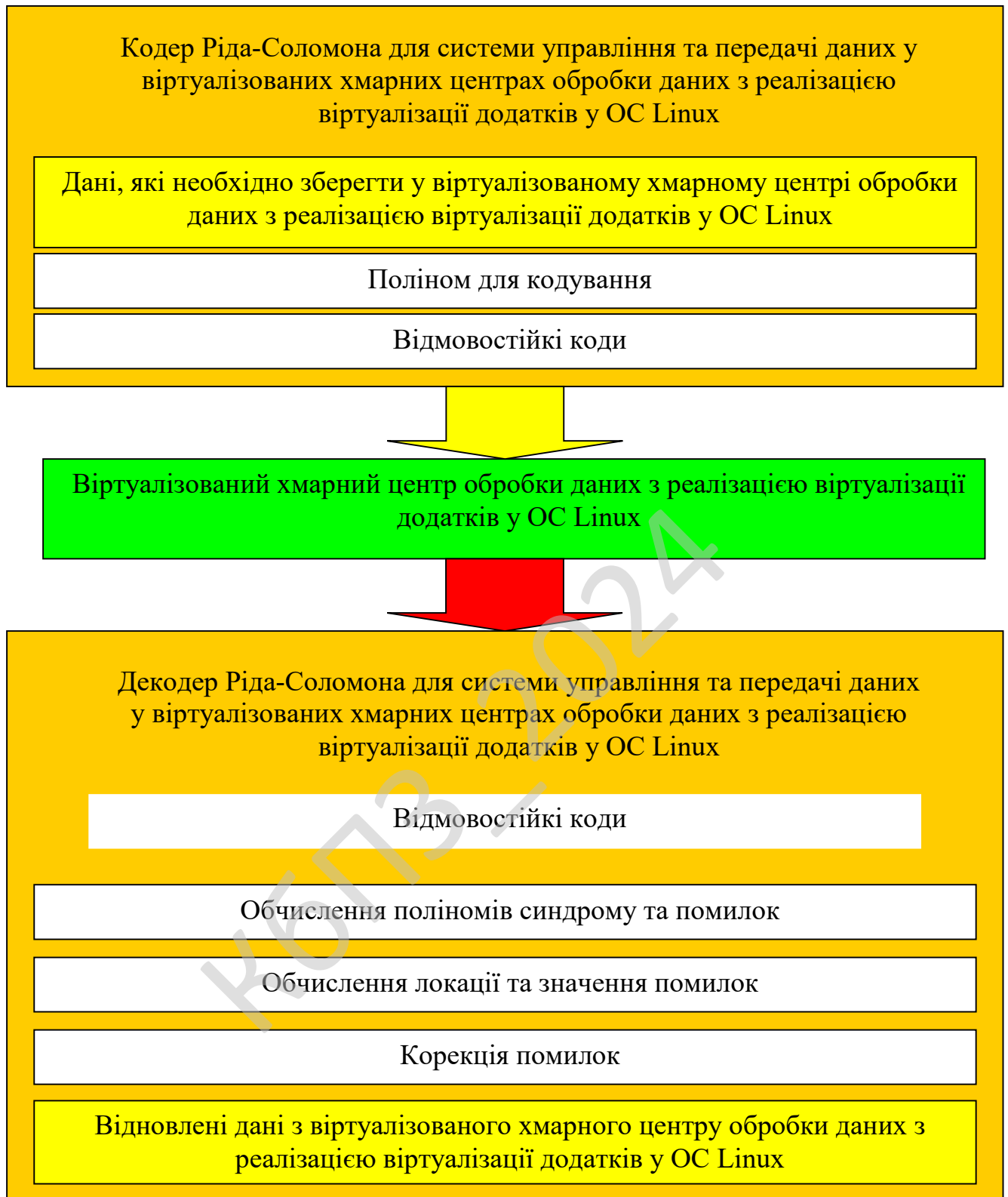


Рисунок 3.2 – Функціональна схема системи

Коди Ріда-Соломона базуються на спеціальному розділі математики – полях Галуа (GF) або кінцевих полях. Арифметичні дії (+, -, x, / і т.д.) над

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

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

Обчислення синдрому

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

Знаходження позицій символічних помилок

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

1. Визначення полінома локації помилок

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

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

Знаходження значень символічних помилок

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

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

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.

КБПЗ – 2024

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		36

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є основою ПЗ. Тому від точності і детальності проробки 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, також те, що при розробці програми слід надати особливу увагу модулю віртуалізації додатків у ОС Linux.

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

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

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

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

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		37

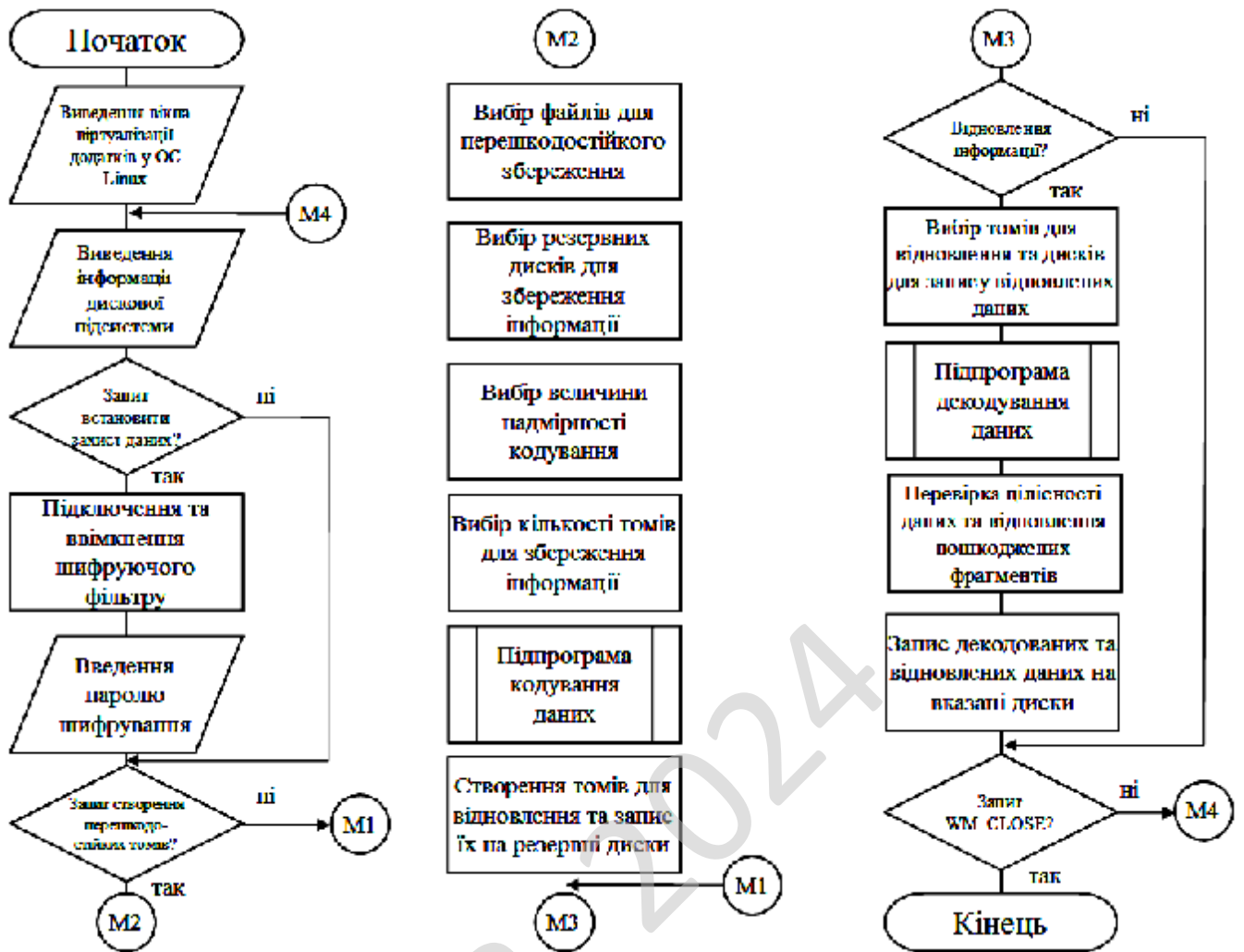


Рисунок 4.1 – Блок-схема основної програми

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

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.

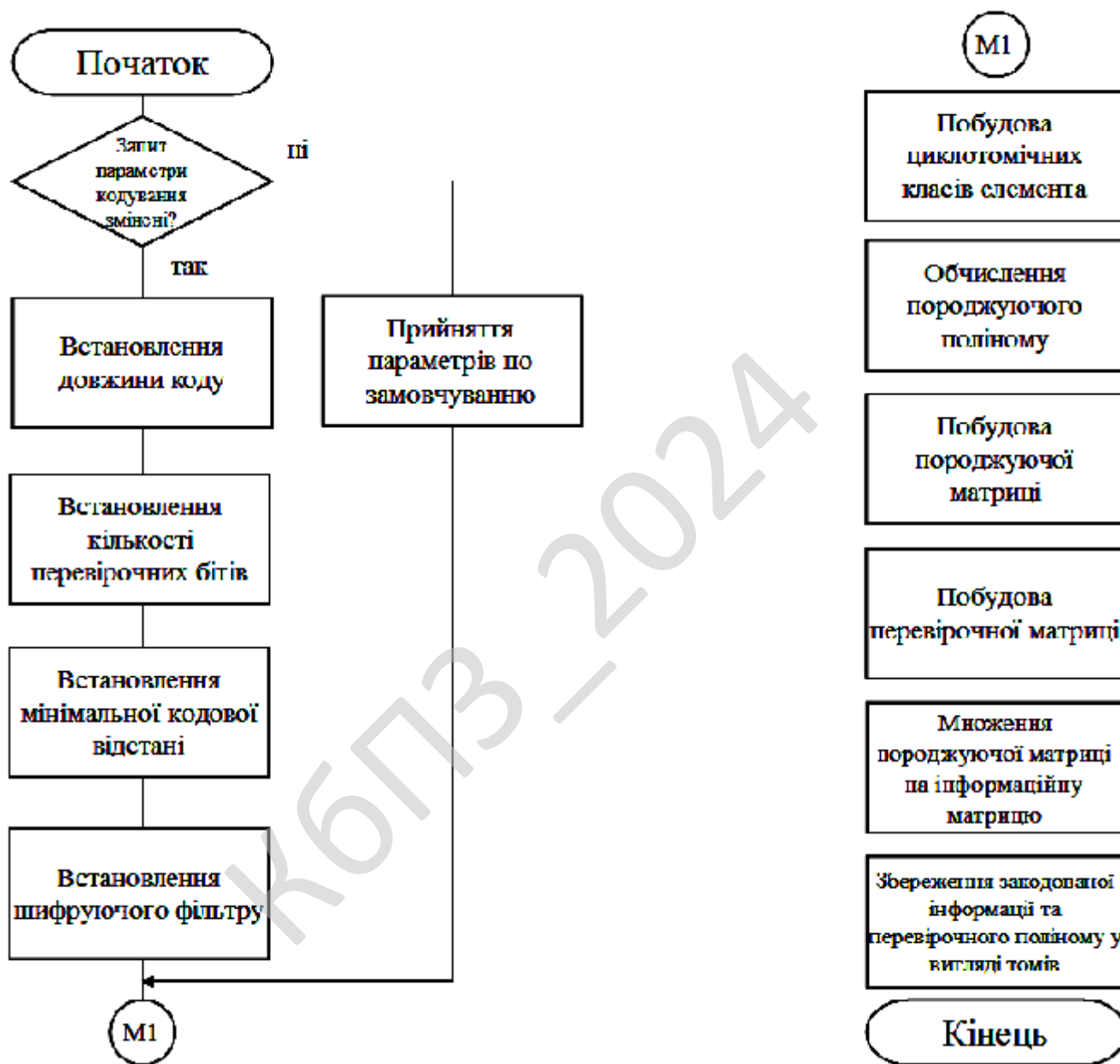


Рисунок 4.2 – Блок-схема роботи підпрограми

Розглянемо використані технології та їх основні компоненти що підтверджують правильність використаних проектних рішень.

Jira – була використана комерційна система відслідковування помилок, призначена для організації взаємодії з користувачами, хоча в деяких випадках використовується і для управління проектами. Розроблено компанією Atlassian, є одним з двох її основних продуктів (поряд з вікі-системою Confluence). Має веб-інтерфейс.

Назва системи отримано шляхом усічення слова «Gojira» – Японського імені монстра Годзилла, що, в свою чергу, є відсиланням до назви конкуруючого продукту – Bugzilla; створювалася в якості заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проектами. Для кожного з проектів створює і веде схеми безпеки і схеми оповіщення.

До версії 3.13.5 (включно) розрізнялися редакції Enterprise, Professional і Standard, після – Залишилася тільки редакція Enterprise (для великих організацій).

Система заснована на Java EE і працює на кількох популярних системах управління базами даних і операційних системах.

Основний елемент обліку в системі – завдання (ticket або issue). Завдання містить назву проекту, тему, тип, пріоритет, компоненти і зміст. Завдання може бути розширена додатковими полями (також і нові призначені для користувача поля можуть бути визначені), додатками (наприклад – Фотографіями, скріншотами) або коментарями. Завдання може редагуватися або просто змінювати статус, наприклад, з «відкритий» в «закритий». Які переходи між станами можливі, визначається через настраюється потік операцій. Будь-які зміни в задачі записуються в журнал.

Jira має велику кількість можливостей конфігурації: для кожної програми може бути визначений окремий тип завдання з власним workflow, набором статусів, одним або декількома видами уявлення (screens). Крім того, за допомогою так званих «схем» можна визначити для кожного індивідуального Jira-проекту власні права доступу, поведінку і видимість полів і багато іншого.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40


```

    int s[n - k + 1];
// поліном синдрому помилки
    int elp[n - k + 2][n - k];
// поліном локатора помилки лямда
    int d[n - k + 2];
    int l[n - k + 2];
    int u_lu[n - k + 2],
    int count = 0, syn_error = 0, root[t], loc[t], z[t + 1],
        err[n], reg[t + 1];
// переводимо отримане кодове слово в індексну форму
// для спрощення обчислень
    for (i = 0; i < n; i++) recd[i] = index_of[recd[i]];
// обчислюємо синдром
//-----
    for (i = 1; i <= n - k; i++)
    {
        s[i] = 0;
// ініціалізація s-регістра
// на його вхід за замовчуванням надходить нуль
        for (j = 0; j < n; j++) if (recd[j] != -1) s[i] ^=
            alpha_to[(recd[j] + i * j) % n];
        if (s[i] != 0) syn_error = 1;
// якщо синдром не дорівнює нулю, зводимо прапор помилки
// перетворимо синдром з поліноміальної форми в індексну
        s[i] = index_of[s[i]];
    }
// корекція помилок
    if (syn_error)
// якщо є помилки, намагаємося їх скорегувати
    {
        // обчислення полінома локатора лямбла
// ініціалізація елементи таблиці
        d[0] = 0; // індексна форма
        d[1] = s[1]; // індексна форма
        elp[0][0] = 0; // індексна форма
        elp[1][0] = 1; // поліноміальна форма
        for (i = 1; i < n - k; i++)
        {
            elp[0][i] = -1;
// індексна форма
            elp[1][i] = 0;
// поліноміальна форма

```

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		42

```

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

```

```

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

```

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		44

```

        root[count] = i;
        loc[count] = n - i;
        count++;
    }
}
if (count == l[u])
{
// немає корінь - ступінь elp < t помилок
// формуємо поліном z(x)
    for (i = 1; i <= l[u]; i++)
// Z[0] завжди дорівнює 1
    {
        if ((s[i] != -1) && (elp[u][i] != -1))
            z[i] = alpha_to[s[i]] ^ alpha_to[elp[u][i]];
        else
            if ((s[i] != -1) && (elp[u][i] == -1))
                z[i] = alpha_to[s[i]];
            else
                if ((s[i] == -1) && (elp[u][i] != -1))
                    z[i] = alpha_to[elp[u][i]];
                else
                    z[i] = 0 ;

        for (j = 1; j < i; j++)
            if ((s[j] != -1) && (elp[u][i - j] != -1))
                z[i] ^= alpha_to[(elp[u][i - j] + s[j]) % n];
// переводимо z[i] в індексну форму
        z[i] = index_of[z[i]];
    }
// обчислення значення помилок у позиціях loc[i]
//-----
    for (i = 0; i < n; i++)
    {
        err[i] = 0;
// переводимо recd[] у поліноміальну форму
        if (recd[i] != -1) recd[i] = alpha_to[recd[i]]; else recd[i] = 0;
    }
// спочатку обчислюємо чисельник помилки
    for (i = 0; i < l[u]; i++)
    {
        err[loc[i]] = 1;
        for (j = 1; j <= l[u]; j++)
            if (z[j] != -1)

```



```
for (j = n - k - 1; j > 0; j--){ b[j] = b[j - 1]; b[0] = 0;
}}}
```

Була використана водоспадна (каскадна) модель життєвого циклу ПЗ (waterfall model) – послідовний метод розробки програмного забезпечення, названий так через діаграму схожу на водоспад.

Ця модель розробки запозичена з системної інженерії у виробництві та будівництві – областях, в яких зміни на пізніх етапах дуже дорогі, або неможливі. Наприклад, для створення складних інженерних конструкцій (споруд, літаків, мостів і т.п.). Зміни в проекті фундаменту будинку після того, як покладений дах коштують дуже дорого, тому перфекціонізм на початкових етапах проектування просто необхідний. Інженери, які починали займатись розробкою програмного забезпечення перейшовши з інших галузей, просто адаптували звичну модель, тому що на ранніх етапах розвитку комп'ютерної техніки не було методологій створених саме для програмування. Проте, схожі методології застосовуються для програмного забезпечення й далі, у випадках коли вимоги фіксовані, і вимагається висока якість та надійність, наприклад в системах для військових чи медичних потреб.

Перший формальний опис водоспадної моделі, після якої вона стала популярною був здійснений В. В. Ройсом у 1970. Попри те, що стаття містить переважно критику методу, на неї часто посилаються.

Переваги методу:

- Ніяких переробок.
- Гарна специфікація перетікає в гарну документацію.
- Зрозуміла модель.
- Розробники можуть мати низьку кваліфікацію.

Недоліки:

- Необхідний перфекціонізм на кожному етапі.
- Важко вносити зміни (якщо взагалі можливо).
- Надлишкове проектування.
- Поділ розробників на "perfect" та "code monkeys".

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		48

Модифікації. Через те що цей метод погано підходить для розробки саме ПЗ, частіше використовують його модифікації.

Найвідоміша модифікація – Sashimi. Названа так через японську страву сашімі (суші нарізане і сервіроване так, що складені рядочком шматочки накладаються один на одного).

В моделі розробки Сашімі фази життєвого циклу йдуть одна за одною, але при цьому перекриваються одна з одною в часі.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою IDEA – симетричний блоковий алгоритм шифрування даних, запатентований швейцарською фірмою Ascom. Відомий тим, що застосовувався в пакеті програм шифрування PGP. У листопаді 2000 року IDEA був представлений як кандидат у проєкті NESSIE в рамках програми Європейської комісії IST (англ. Information Societes Technology, інформаційні громадські технології).

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

Так як IDEA використовує 128-бітний ключ і 64-бітний розмір блоку, відкритий текст розбивається на блоки по 64 біт. Якщо таке розбиття неможливо, останній блок доповнюється різними способами певною послідовністю біт. Для

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		49

уникнення витоку інформації про кожному окремому блоці використовуються різні режими шифрування. Кожен вихідний незашифрований 64 – біт ний блок ділиться на чотири підблока по 16 біт кожен, так як всі алгебраїчні операції, що використовуються в процесі шифрування, відбуваються над 16-бітними числами. Для шифрування і розшифрування IDEA використовує один і той же алгоритм.

Позначення операцій:

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

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

Додавання за модулем 2^{16} .

Множення за модулем $2^{16}+1$.

Побітова виключна диз'юнкція (XOR).

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

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

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

Генерація ключів

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

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		50

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

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

– Додавання за модулем 2^{16} .

– Побітове виключне АБО.

В кінці кожного раунду шифрування є чотири 16-бітних підблоки, які потім використовуються як вхідні підблоки для наступного раунду шифрування. Вихідна перетворення являє собою скорочений раунд, а саме, чотири 16-бітних підблоки на виході восьмого раунду і чотири відповідних підключа піддаються операціям:

– Множення за модулем $2^{16}+1$.

– Додавання за модулем 2^{16} .

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

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		52

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ віртуалізації додатків у ОС Linux яке зображено на рисунку 5.1. Розрахунок проводиться через консоль, а результати передаються до графічного інтерфейсу. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

- Навігаційне меню: Файл; Дата-центр; Збереження; Відновлення; Тестування; Параметри.
- Вікна обрання групи .
- Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.
- Функціональних кнопок ПЗ.

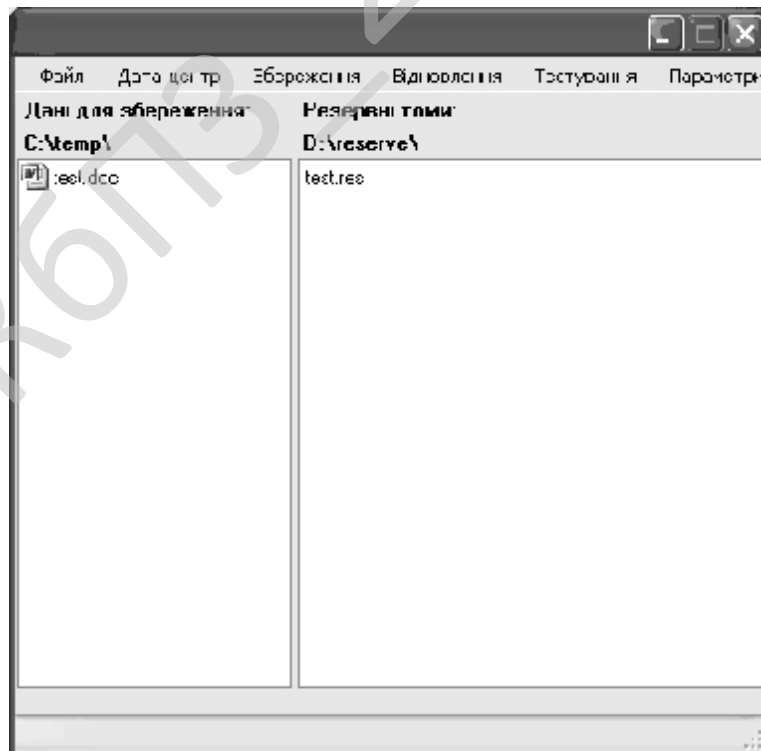


Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Linux без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

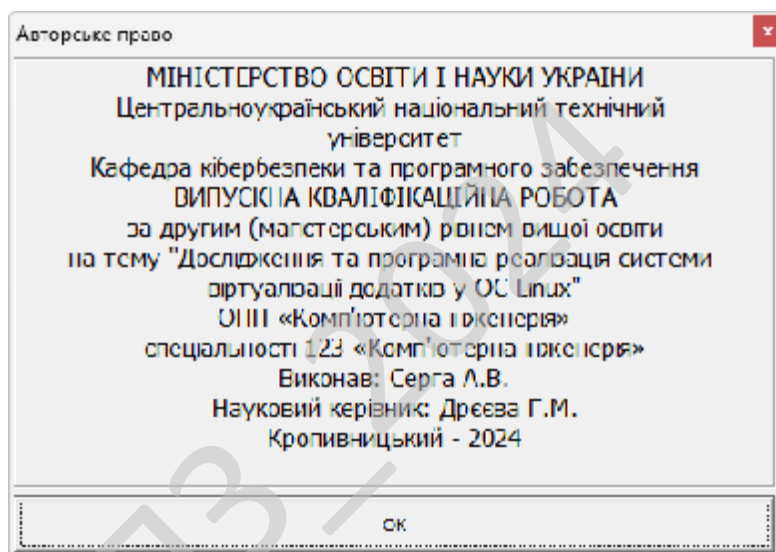


Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		54

унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

- Випуск.
- Встановлення та активація.
- Деактивація.
- Адаптація.
- Обновлення.
- Вмонтування.
- Відстежування версій.
- Видалення.
- Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		55

в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

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.

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

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		56

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.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		57

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віртуалізації додатків у ОС Linux.

Метою розробки є дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux.

Об'єктом дослідження є процес віртуалізації додатків у ОС Linux.

Предметом дослідження є методи віртуалізації додатків у ОС Linux.

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

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

- Удосконалено метод віртуалізації додатків у ОС Linux.
- Розроблено вітчизняний продукт віртуалізації додатків у ОС Linux, який має більш широкі можливості, на відміну від існуючих аналогів.

					VKPM-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		58

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та програмної реалізації системи віртуалізації додатків у ОС Linux можуть бути цікавими для різних категорій осіб та організацій (рис. 7.1).

1. Розробники програмного забезпечення	Інженери-програмісти, які працюють над створенням та оптимізацією додатків, можуть використовувати результати для поліпшення продуктивності та зручності розробки.
2. Системні адміністратори	Спеціалісти з адміністрування, які відповідають за управління серверами та додатками, можуть знайти результати корисними для оптимізації інфраструктури та підвищення її надійності.
3. Фахівці з DevOps	Команди DevOps можуть використовувати результати для інтеграції віртуалізації в CI/CD процес (безперервна інтеграція та безперервне постачання).
4. Компанії, що займаються ІТ-консалтингом	Консалтингові фірми можуть впроваджувати отримані знання у проекти для клієнтів, пропонуючи рішення для віртуалізації додатків.
5. Виробники апаратного забезпечення	Компанії, які виробляють сервери та мережеве обладнання, можуть використовувати результати для оптимізації своїх продуктів під нові технології віртуалізації.
6. Наукові установи та дослідницькі організації	Дослідники, які займаються питаннями віртуалізації, можуть використати результати для подальшого розвитку теорії та практики в цій галузі.
7. Стартупи у сфері інформаційних технологій	Стартупи, які прагнуть розробити нові рішення для віртуалізації, можуть використовувати результати для вдосконалення своїх продуктів.
8. Освітні заклади	Університети та коледжі, які пропонують курси з комп'ютерних наук, можуть інтегрувати результати в свої навчальні програми.
9. Клієнти, які використовують віртуалізацію	Компанії, що впроваджують віртуалізацію для підвищення ефективності, можуть скористатися отриманими знаннями для оптимізації своїх процесів.
10. Експерти з безпеки	Спеціалісти з кібербезпеки можуть бути зацікавлені в результатах дослідження, щоб оцінити ризики та переваги використання віртуалізації у контексті безпеки.

Рисунок 7.1 – Цільова аудиторія

Отже, результати дослідження та реалізації системи віртуалізації додатків у ОС Linux можуть бути цінними для широкого кола фахівців, які зацікавлені у підвищенні ефективності, продуктивності та безпеки віртуалізованих середовищ.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Оцінка привабливості програмної реалізації системи віртуалізації додатків у ОС Linux шляхом застосування методів експертних оцінок може бути виконана у кілька етапів. Перед початком оцінки важливо визначити ключові критерії, за якими буде оцінюватися система (рис. 7.2).

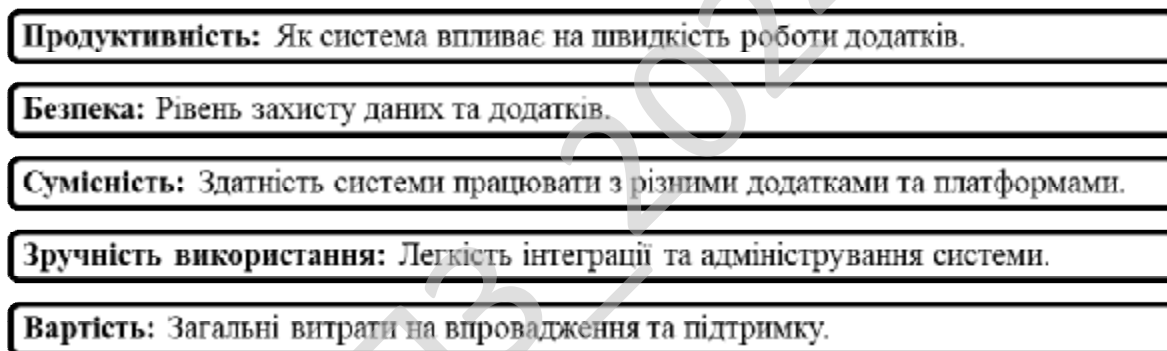


Рисунок 7.2 – Критерії експертної оцінки

Залучаємо групу експертів у сфері віртуалізації, розробки програмного забезпечення та системного адміністрування для оцінки зазначених критеріїв. Експерти можуть представляти різні компанії або організації, які мають досвід у цій галузі.

Обираємо такий метод експертних оцінок як метод анкетування, де експерти оцінюють кожен критерій за шкалою від 1 до 10, де 1 – дуже низький, а 10 – дуже високий.

Проводимо анкетування серед експертів.

Таблиця 7.1 – Зведені результати експертних оцінок

Критерій	Експерт 1	Експерт 2	Експерт 3	Середнє значення
Продуктивність	8	7	9	8.0
Безпека	9	8	8	8.3
Сумісність	7	7	6	6.7
Зручність	8	9	8	8.3
Вартість	6	7	5	6.0

Після збору оцінок обчисліть середнє значення для кожного критерію, щоб отримати загальну картину привабливості системи. У нашому випадку: продуктивність: 8.0, безпека: 8.3, сумісність: 6.7, зручність: 8.3, вартість: 6.0.

На основі отриманих оцінок експертів можна зробити висновок про загальну привабливість програмної реалізації системи віртуалізації додатків у ОС Linux. Наприклад, висока оцінка за критеріями продуктивності, безпеки та зручності використання свідчить про те, що система має хороший потенціал на ринку, тоді як нижчі оцінки за сумісністю та вартістю можуть вимагати додаткових досліджень та вдосконалень. На основі аналізу результатів експертної оцінки можна розробити рекомендації щодо покращення системи. Це може включати оптимізацію витрат, покращення сумісності з іншими системами, або вдосконалення інтерфейсу для підвищення зручності використання.

7.3 Вибір методу оцінки вартості ПЗ

Вибір методу оцінки вартості для програмної реалізації системи віртуалізації додатків у ОС Linux залежить від конкретних умов проекту, доступних даних та цілей оцінки. Якщо у вас є дані про попередні подібні проекти, метод аналогій або експертних оцінок може бути дуже корисним. Для більш точного планування бюджету підійдуть методи FPA або Bottom-Up. Якщо

ви хочете врахувати ризики, використовуйте метод трьох оцінок. Найкращий метод оцінки вартості для програмної реалізації системи віртуалізації додатків у ОС Linux залежить від ваших конкретних потреб, доступних даних і бажаної точності оцінки. Рекомендується також комбінувати кілька методів для отримання більш об'єктивної оцінки.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Економічна ефективність впровадження системи віртуалізації додатків у ОС Linux може бути оцінена за допомогою різних показників. Наші розрахунки ілюструють, як це може виглядати.

Припустимо, компанія вирішила впровадити систему віртуалізації для оптимізації своєї ІТ-інфраструктури, яка включає 50 фізичних серверів, що виконують різноманітні завдання, такі як хостинг веб-додатків, бази даних і файлові системи (таблиця 7.2).

При цьому врахуємо витрати на впровадження системи віртуалізації: вартість програмного забезпечення: 50,000, вартість навчання персоналу: 10,000, загальні витрати на впровадження: 60,000.

Розрахунок чистої економічної ефективності

Загальна економічна вигода за рік = 988,000. Загальні витрати на впровадження = 60,000. Чистий прибуток=988,000–60,000=928,000 грн.

Таким чином, впровадження системи віртуалізації додатків у ОС Linux приносить значну економічну ефективність, що підтверджується скороченням витрат на апаратуру, електроенергію, обслуговування та збільшенням доходу. В загальному, компанія може очікувати значну вигоду від впровадження віртуалізації, яка може перевищити 900,000 у перший рік після впровадження.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		62

Таблиця 7.3 – Зведені результати загальної економічної ефективності

Показник	Сума
Економія на апаратному забезпеченні	800,000
Економія на електроенергії	48,000
Економія на обслуговуванні	40,000
Збільшення доходу	100,000
Загальна вигода	988,000

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Алгоритм просування проєкту програмної реалізації системи віртуалізації додатків у ОС Linux візуалізовано на рисунок 7.3.

Цей алгоритм просування проєкту програмної реалізації системи віртуалізації додатків у ОС Linux забезпечить структурований підхід до виходу на ринок, залучення клієнтів та підвищення впізнаваності бренду.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації каналів збуту та шляхів реалізації проєкту програмної реалізації системи віртуалізації додатків у ОС Linux можна застосувати кілька стратегій:

- **сегментація клієнтів:** визначте специфічні сегменти ринку, які найбільше виграють від вашої системи віртуалізації (наприклад, малий та середній бізнес, освітні установи, стартапи в ІТ).
- **індивідуальні пропозиції:** створіть пропозиції, які відповідають потребам кожного сегмента.
- **веб-сайт та електронна комерція:** створіть зрозумілий і зручний веб-сайт з можливістю покупки та завантаження програми безпосередньо.

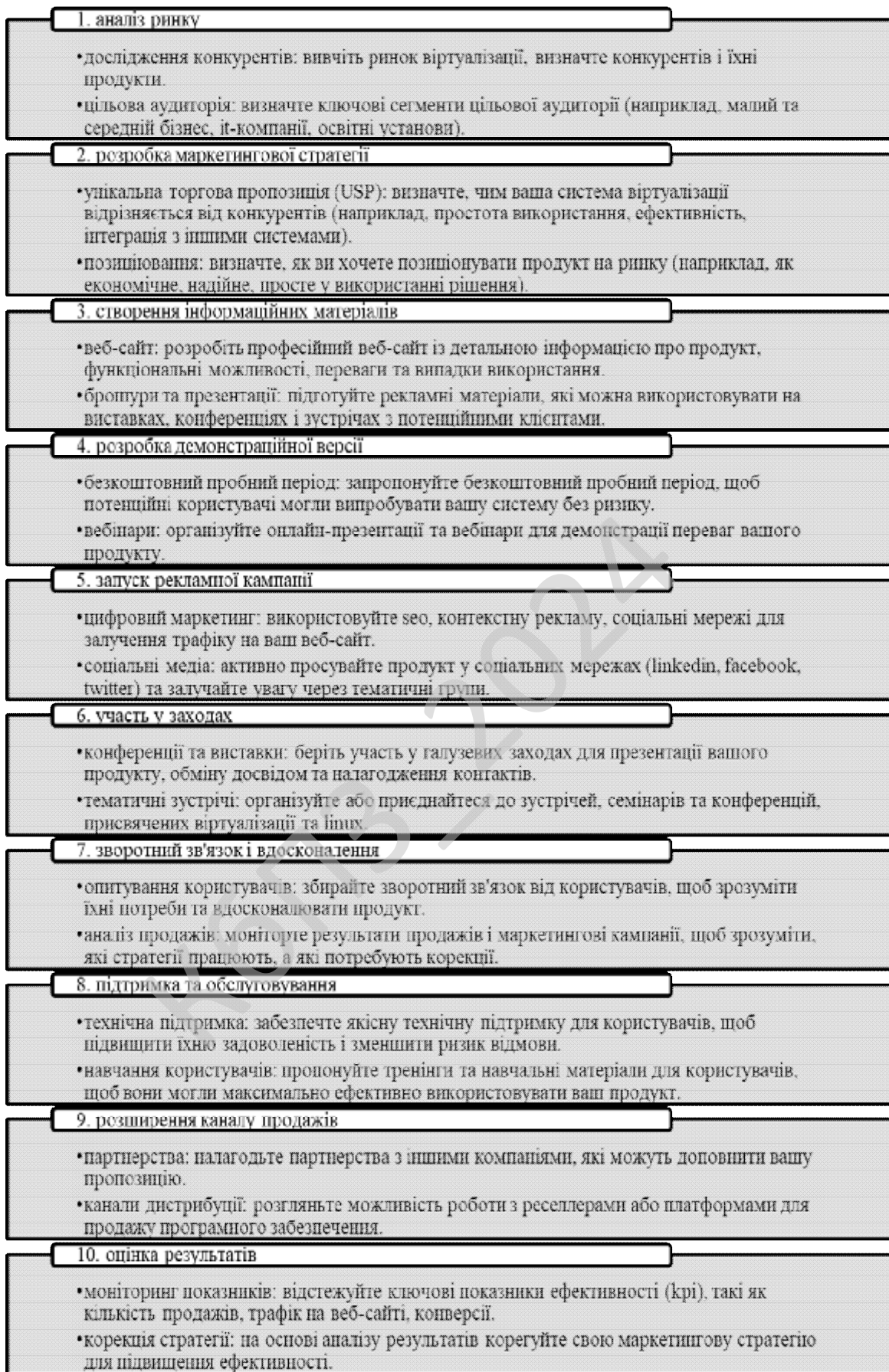


Рисунок 7.3 – Алгоритм просування проєкту

– **SEO-оптимізація:** оптимізуйте контент для пошукових систем, щоб залучити органічний трафік на ваш веб-сайт.

Оптимізація каналів збуту та шляхів реалізації проєкту програмної реалізації системи віртуалізації додатків у ОС Linux вимагає комплексного підходу. Зосередьтеся на аналізі ринку, створенні партнерств, ефективному використанні онлайн-каналів і наданні підтримки користувачам. Це допоможе залучити нових клієнтів, підвищити їхню лояльність і, в результаті, збільшити обсяги продажів.

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключові фактори успіху проєкту програмної реалізації системи віртуалізації додатків у ОС Linux можуть включати наступні елементи – рисунок 7.4.

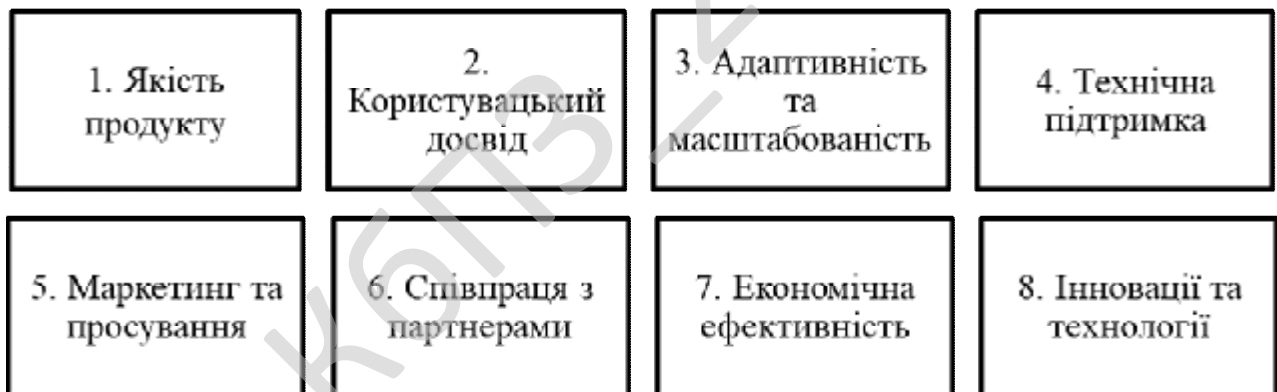


Рисунок 7.4 – Ключові фактори успіху проєкту

Успіх проєкту програмної реалізації системи віртуалізації додатків у ОС Linux залежить від багатьох факторів, починаючи з якості продукту і закінчуючи маркетинговими стратегіями. Зосередження на цих ключових аспектах може суттєво підвищити шанси на успіх і задоволеність клієнтів.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Електронно-обчислювальна машина (ЕОМ) відіграє важливу роль у житті сучасної людини. Кожного дня мільйони людей використовують ЕОМ для пошуку необхідної інформації, спілкуванні у соціальних мережах, перегляду новин, роботи тощо. Багато людей користуються ЕОМ у професійних цілях, оскільки завдяки ЕОМ з'явилося багато нових професій. Тому для розробника хмарних сервісів так важливо розробити зручний інтерфейс для зручного сприйняття інформації, та необхідний функціонал, який буде відповідати необхідним вимогам та навантаженням. Все це вимагає багато часу та великого навантаження з боку розробників. Тому так важливо слідкувати за умовами праці, в яких відбувається робочий процес. Оскільки захворювання можуть бути спричинені надмірним фізичним або розумовим навантаженням, через велику нервово-емоційну напругу, або через виробниче середовище. В даному розділі магістерської роботи проведемо аналіз основних чинників при роботі програміста.

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

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		68

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машина (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, параметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

На роботу програміста впливають наступні фактори: невідповідний мікроклімат приміщення (температура, вологість), недостатня освітленість робочої зони, підвищений рівень шуму та електромагнітного випромінювання, порушення іонного складу повітря, неправильна ергономічна організація робочого місця, ризики, пов'язані із погіршенням зору, порушенням фізичного стану, стресом тощо.

Шкідливими факторами при роботі з персональним комп'ютером є неіонізуюче випромінювання промислової частоти, збільшене нервово-емоційне навантаження на оператора, збільшення навантаження на органи зору та дрібні стереостатичні рухи кінцівок.

Ці фактори можуть викликати у працівника певні розлади здоров'я, зокрема підвищення артеріального тиску, кон'юктивіти, тендовагініти та інші захворювання. Комп'ютер, як і будь-який електричний прилад, особливо при його неправильному підключенні, може бути джерелом ураження оператора електричним струмом. Саме тому всі працівники, які працюють з персональним комп'ютером, повинні мати першу (або другу) групу допуску з електробезпеки.

Через наявність зазначених факторів працівники, які працюють з персональними комп'ютерами, підлягають попередньому та періодичному медичному огляду згідно з пунктом 6.2.3 додатку 4 до наказу Міністерства охорони здоров'я України «Про затвердження Порядку проведення медичних оглядів працівників певних категорій» від 21 травня 2007 року № 246.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		69

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Оптимальна температура в приміщенні для праці має становити 20 -24 °С, відносна вологість – 40 – 60 %, атмосферний тиск – 750 мм.рт.ст., запиленість не повинна перевищувати 10 мг/м³, швидкість руху повітря – 0,1 м/с.

Через те, що обчислювальна техніка є джерелом тепловиділення, організація мікроклімату потребує додаткових зусиль: кондиціонування, провітрювання, використання систем опалення тощо. Об'єм приміщень повинен передбачатися з урахуванням як мінімум 20 м³ /на особу [4].

Монітори комп'ютерів є джерелом випромінювання, яке може зашкодити здоров'ю людини. Для забезпечення роботи з комп'ютером відстань від монітора повинна становити не менше 50 см, бажано використовувати монітори зі зниженим рівнем, скорочувати час безперервної роботи за комп'ютером (робити п'ятнадцяти хвилинні перерви після кожних півтори години праці). Також в приміщенні необхідно встановлювати іонізатори повітря, використовувати нейтралізатори та зволожувачі.

Комп'ютери та периферійні пристрої є джерелами шуму, висока інтенсивність якого може призвести до проблем з органами слуху та негативно впливати на психологічний стан. Рівень шуму на робочому місці не повинен перевищувати 50 дБА [5]. Для зменшення рівня шуму можна використовувати звукопоглинальні пристрої, а стіни приміщень з комп'ютерами можуть бути покриті звукопоглинальними матеріалами. Поряд із шумом часто виникає вібрація. Для зменшення рівня вібрації в приміщенні на поверхні необхідно встановлювати віброізолятори.

Ергономічні показники робочого місця програміста мають бути наступними: висота робочої поверхні повинна складати 720 мм, розмір поверхні має становити 1600 x 1000 мм; під столом повинен бути простір з розмірами по глибині 650 мм; стіл повинен мати підставку для ніг, розташовану під кутом

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		70

15° до поверхні; відстань клавіатури від краю столу має бути не більше 300 мм; відстань між очима й екраном повинна складати 40 – 80 см; стілець повинен мати підйомно-поворотний механізм; висота сидіння має регулюватися в межах 400 – 500 мм, глибина – не менше 380 мм, а ширина – не менше 400 мм, висота опорної поверхні спинки має бути не менше 300 мм, ширина – не менше 380 мм. Кут нахилу спинки стільця до площини сидіння повинен змінюватися в межах 90 – 110° [6].

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

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер Prinics PicKit M1 Smartphone Photo Printer White, електродвигуни вентиляторів ЕОМ.

Робота програміста передбачає постійний візуальний контакт з моніторами комп'ютерів, та, як наслідок, значне навантаження на зір. Традиційно, це зорова робота високої або середньої точності. Для зорової роботи високої точності загальне освітлення (розподіл світла у всьому об'ємі приміщення) має становити 300 лк, комбіноване освітлення (поєднання загального і місцевого освітлення) – 750 лк. Штучне освітлення повинно бути рівномірним та використовуватися в світлий і темний час доби. Джерелами штучного освітлення можуть слугувати люмінесцентні лампи. Правильне освітлення передбачає уникнення відблисків на екранах.

З 2019 року діють Державні будівельні норми України “Природне і штучне освітлення” – ДБН В.2.5-28:2018 [4], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [4], можна віднести до роботи з малою точністю (найменший

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		71

розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи B). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [4], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

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

- розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;
- мікроклімат відповідає нормативному значенню;
- акустичні умови роботи не перевищують нормативних значень;

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

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		72

нашому випадку $K = 1,5$);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку $Z = 1,1$);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці [8]; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін} = 50\%$ і $\rho_{стелі} = 50\%$.

Обчислимо індекс приміщення за формулою:

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

де: S – площа приміщення, $S = 42 \text{ м}^2$;

h – розрахункова висота підвісу, $h = 2,9 \text{ м}$ (співпадає з висотою стелі, т.я. лампи освітлення закріплюються на стелі);

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

B – довжина приміщення, $B = 7 \text{ м}$.

Підставимо всі значення у формулу та визначимо індекса приміщення:
 $i = 1,4$.

Знаючи індекс приміщення, за знаходимо $n = 0,29$ (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам) [8]. Підставимо всі значення у формулу, визначимо світловий потік: $F = 71689 \text{ Лм}$.

Для розрахунку дудемо використовувати світлодіодні стельові панелі *Delux LED Panel 41 44Wm.*, світловий потік яких $F_{л} = 3600 \text{ Лм}$.

Число ламп визначається по формулі:

$$N = F / F_{л}$$

де: F – світловий потік,

$F_{л}$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначимо індекса приміщення:

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		74

$$N = 71689 / 3600 = 19,9 \text{ шт.}$$

Приймаємо необхідну кількість світлодіодних світильників 20 шт.

Висновки до розділу

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

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

КБПЗ - 2024

					VKPM-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		75

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи віртуалізації додатків у ОС Linux.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів віртуалізації додатків у ОС Linux.

Рішення даного завдання полягало у вирішенні наступних задач:

- Був проведений огляд існуючих систем віртуалізації додатків у ОС Linux.
- Досліджена система віртуалізації додатків у ОС Linux.
- На основі отриманих результатів досліджень створена програмна реалізація системи віртуалізації додатків у ОС Linux.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання віртуалізації додатків у ОС Linux.

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

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

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		76

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

Програма реалізована на мові високого рівня Visual C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Linux.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм IDEA.

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

Проведено маркетингове та економічне обґрунтування IT-проєкту, що дозволило визначити ключові фактори успіху даного проєкту.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		77

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Серга А.В. Дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2024.

2. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.

3. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.

4. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

5. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

6. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

7. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

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

9. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		78

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

11. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

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

13. Smirnov, O., Kuznetsov, A., 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.

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

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

16. Smirnov, O., 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.

17. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated

with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

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

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

26. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка*. 2024. №3(23), С. 111-131.

27. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

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

29. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція «Інформаційна безпека та комп'ютерні технології», м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

30. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		81

підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

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

32. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

33. Смірнов О.А., Дреєва Г.М., Дреєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральнуукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

34. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

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

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

37. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		82

38. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

39. Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

40. Smirnov Oleksii, Kovalenko Oleksandr, Kovalenko Anna, Smirnov Serhii. Method of testing the DOM XSS vulnerability. International Conference «Information technologies, systems and networks ITSН-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. 2017. P7.

41. Смірнов О.А., Смірнов С.А., Коваленко О.В., Коваленко А.С. Технологія тестування DOM XSS уразливості. Науково-практичний журнал кібер безпеки (SPCSJ) № 1. [Електронний журнал]. Грузія. Тбілісі: SCSA – 2017.

42. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

43. Смірнов О.А., Смірнов С.А., Рябой Д.К., Рябая О.В. Модель вузла комутації з відносними пріоритетами, резервуванням ресурсів і обліком реальної надійності обслуговуючих приладів. Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

44. Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		83

програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

45. Смірнов О.А., Лисенко І.А. Формалізація процесу проектування тестових наборів. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 3 (48). – Харків: ХУПС. – 2016. – С.96-100.

46. Смірнов О.А., Лисенко І.А. Удосконалення методу перевірки коректності таблиць рішень для подання тестових наборів. Збірник наукових праць "Системи обробки інформації". – Випуск 8 (145). – Х.: ХУПС – 2016. – С. 77-80.

47. Смірнов О.А., Лисенко І.А. Розробка впорядкованих каскадних таблиць рішень із використанням матриць слідування. Збірник наукових праць "Системи обробки інформації". – Випуск 6 (143). – Х.: ХУПС – 2016. – С. 216-220.

48. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод кількісної оцінки ризиків розроблення програмного забезпечення. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 128-133.

49. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод якісного аналізу ризиків розроблення програмного забезпечення. Наука і техніка Збройних Сил України. – Випуск 2(23). – Харків: ХУПС. – 2016. – С. 150-158.

50. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Проблеми аналізу та оцінки ризиків інформаційної діяльності. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 40-42.

51. Смірнов О.А., Коваленко А.С., Коваленко О.В., Доренський О.П. Удосконалення методу технічного обслуговування об'єктів інтегрованої інформаційної системи. Системи озброєння і військова техніка. – Випуск 2(46) – Х.: ХУПС – 2016. – С. 103-107.

					ВКРМ-123.24.0039.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		84

Додаток А
(обов'язковий)

Технічне завдання

Зміст

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

					ВКРМ-123.24.0039.00.00.ТЗ			
Вим.	Арк.	№ документа	Підпис	Дата				
Розробив	Серга А.В.				Дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux	Літ.	Аркуш	Аркушів
Перевірів	Дресва Г.М.					М	1	6
Н. Контр.	Коваленко А.С.				ЦНТУ КІ-23М			
Затв.	Смірнов О.А.							

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи віртуалізації додатків у ОС Linux.

2 Підстава для розробки

Підставою для розробки служить завдання на випускню кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 07.08.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи віртуалізації додатків у ОС Linux.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

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

					ВКРМ-123.24.0039.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		2

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

5.2 Показники призначення

Система повинна забезпечувати:

- програмну реалізацію системи віртуалізації додатків у ОС Linux;
- цілісність даних у процесі роботи та при зберіганні;
- простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

					ВКРМ-123.24.0039.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		3

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

- температура повітря: 19-20 град. по Цельсію;
- відносна вологість повітря до 80%;
- атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Linux і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Visual C#.

					ВКРМ-123.24.0039.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		2

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

7.2 Виконати маркетингове та економічне обґрунтування ІТ-проєкту з урахуванням цін на 3 вересня 2024 року.

8 Вимоги щодо охорони праці

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

					ВКРМ-123.24.0039.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		5

9 Перелік документів, що розробляються

- Наукова новизна – 1 аркуш.
- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Показники економічної ефективності – 1 аркуш.
- Пояснювальна записка – 84 аркуші.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Маркетингове та економічне обґрунтування ІТ-проєкту.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 02.12.2024 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 16.12.2024 р.

					ВКРМ-123.24.0039.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		6

Додаток Б
(обов'язковий)

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за
другим (магістерським) рівнем вищої освіти

_____ Дреєва Г.М.

*Дослідження та програмна реалізація
системи віртуалізації додатків у ОС Linux*

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 56

Літера: РП

Кропивницький – 2024 року

Файл ProcessForm.cs - вікно відображення процесів запису/читання та перевірки цілісності даних при віртуалізації додатків у ОС Linux

```

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

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

        #region Windows Form Designer generated code

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

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

```

```

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

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

```

```

        this.fileAnalyzeStatGroupBox.Controls.Add(this.percOfAltEccLabel);
        this.fileAnalyzeStatGroupBox.Controls.Add(this.percOfDamageLabel);
        this.fileAnalyzeStatGroupBox.Controls.Add(this.percOfAltEccLabel_);
        this.fileAnalyzeStatGroupBox.Controls.Add(this.percOfDamageLabel_);
        this.fileAnalyzeStatGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
        this.fileAnalyzeStatGroupBox.Location = new System.Drawing.Point(12,
216);

        this.fileAnalyzeStatGroupBox.Name = "fileAnalyzeStatGroupBox";
        this.fileAnalyzeStatGroupBox.Size = new System.Drawing.Size(459,
64);

        this.fileAnalyzeStatGroupBox.TabIndex = 0;
        this.fileAnalyzeStatGroupBox.TabStop = false;
        this.fileAnalyzeStatGroupBox.Text = "Результат аналізу цілісності
даних";

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

```

```

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

```

```

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

```

```

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

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

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

    }

#endregion

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

```

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

K6ПЗ_2024

Файл FileAnalyzer.cs - контроль цілісності даних

```

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

namespace Data_Center
{
    /// <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[] VollList
{
    get
    {
        if (!InProcessing)
        {
            return this.vollList;

        } else
        {
            return null;
        }
    }
}

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

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

```

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

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

/// <summary>
/// Пріоритет процесу
/// </summary>
public int ThreadPriority
{
    get
    {
        return (int)this.threadPriority;
    }
    set
    {
        if (
            (this.thrFileAnalyzer != null)
            &&
            (this.thrFileAnalyzer.IsAlive)
        )
        {
            switch (value)
            {
                default:
                case 0:
                {
                    this.threadPriority =
System.Threading.ThreadPriority.Lowest;

                    break;
                }

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

                    break;
                }

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

                    break;
                }

                case 3:
                {
```

```

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

        break;
    }

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

        break;
    }
}

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

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

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

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

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

#endregion Public Properties & Data

#region Data

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

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

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

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

```

```

    /// </summary>
    private String fileName;

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

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

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

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

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

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

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

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

    #endregion Data

    #region Construction & Destruction

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

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

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

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

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

```

```

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

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

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

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

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

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

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

#endregion Construction & Destruction

#region Public Operations

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

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

        return false;
    }

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

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

    // Зберігаємо тип кодера коду Ріда-Соломона (по типу
    використовуваної матриці кодування)
    this.codecType = codecType;

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

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

        // Повертаємо результат обробки
        return this.processedOK;
    }

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

    //...потім даємо йому ім'я...

```

```

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

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

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

// Повідомляємо, що все нормально
return true;
}

/// <summary>
/// Метод запуску потоку обробки перевірки CRC64, записаного в кінець
/// кожного з файлів набору, з генеруванням списку наявних томів
"vollList",
/// який буде використаний декодером для відновлення даних
/// </summary>
/// <param name="path">Шлях до файлів для обробки</param>
/// <param name="fileName">Ім'я файлу для обробки</param>
/// <param name="dataCount">Конфігурація кількості основних
томів</param>
/// <param name="eccCount">Конфігурація кількості томів для
відновлення</param>
/// <param name="codecType">Тип кодера коду Ріда-Соломона (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)Code_Rid_Solomon_Const.MaxVolCountAlt)
    )
    {
        // Активуємо індикатор актуального стану змінних-членів
        this.finished = true;

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

        return false;
    }

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

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

    // Зберігаємо тип кодека кодера коду Ріда-Соломона (по типу
використовуваної матриці кодування)
    this.codecType = codecType;

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

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

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

        // Повертаємо результат обробки
        return this.processedOK;
    }

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

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

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

```

```

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

        // Повідомляємо, що все нормально
        return true;
    }

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

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

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

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

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

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

    #endregion Public Operations

    #region Private Operations

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

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

        // Піддаємо обробці всі томи
        for (int volNum = 0; volNum < (this.dataCount + this.eccCount);
        volNum++)
        {

```

```

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

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

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

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

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

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

        // Чекаємо кожне з перерахованих подій...
        ManualResetEvent[] { this.wakeUpEvent[0], this.exitEvent[0],
this.eFileIntegrityCheck.FinishedEvent[0] });

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

            continue;
        }

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

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

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

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

            return;
        }
    }
}

```

```

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

        } // while(true)

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

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

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

        return;
    }

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

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

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

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

        return;
    }

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

```

```

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

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

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

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

    return;
}

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

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

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

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

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

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

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

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

    // Індекс у масиві томів
    int volListIdx = 0;

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

    // Лічильник кількості ушкоджених основних томів

```

```

int dataVolMissCount = 0;

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

// Ім'я файлу для обробки
String fileName;

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

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

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

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

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

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

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

                    // Чекаємо кожне з перерахованих подій...
                    int 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
    this.wakeupEvent[0].Reset();

    continue;
}

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

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

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

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

    return;
}

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

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

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

    return;
}

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

```

нас прокинутися

змінних-членів

вкладеним алгоритмом...

(цього й чекали в while(true)!) break;

членів

потоків

```

        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);
}

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

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

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

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

    return;
}

}

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

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

// Тепер, коли знаємо кількість ушкоджених основних томів,

```

```

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

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

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

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

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

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

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

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

                    //...якщо одержали сигнал до того, щоб
                    // переходимо на нову ітерацію, тому що
                    // перед постановкою на паузу...
                    if (eventIdx == 0)
                    {
                        //...попередньо скинувши подію, що змусила
                        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
        this.finished = true;

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

        return;
    }

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

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

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

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

    return;
}

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

// Указуємо, що том для відновлення коректний

```

змінних-членів

вкладеним алгоритмом...

(цього й чекали в while(true)!)

членів

потоків

```

        if (this.eFileIntegrityCheck.ProcessedOK)
        {
            eccVolIsOK = true;
        }

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

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

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

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

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

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

        return;
    }
}

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

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

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

}

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

```

```

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

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

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

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

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

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

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

    return;
}

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

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

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

    return;
}

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

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

```

```
// зупиняючись на коректному томі для відновлення
while (altEccList[altEccListIdx] == -1)
{
    altEccListIdx++;
}

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

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

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

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

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

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

#endregion Private Operations
}
}
```

Файл MainForm.cs - головне вікно програми

```

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

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

        #region Windows Form Designer generated code

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

```

```

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

```

```

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

```

```

        // aboutToolStripMenuItem
        //
        this.aboutToolStripMenuItem.Name = "aboutToolStripMenuItem";
        this.aboutToolStripMenuItem.Size = new System.Drawing.Size(166, 22);
        this.aboutToolStripMenuItem.Text = "Про програму...";
        this.aboutToolStripMenuItem.Click += new
System.EventHandler(this.aboutToolStripMenuItem_Click);
        //
        // coderConfigGroupBox
        //
        this.coderConfigGroupBox.Controls.Add(this.redundancyGroupBox);
        this.coderConfigGroupBox.Controls.Add(this.allVolCountGroupBox);
        this.coderConfigGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
        this.coderConfigGroupBox.Location = new System.Drawing.Point(414,
26);

        this.coderConfigGroupBox.Name = "coderConfigGroupBox";
        this.coderConfigGroupBox.Size = new System.Drawing.Size(561, 98);
        this.coderConfigGroupBox.TabIndex = 5;
        this.coderConfigGroupBox.TabStop = false;
        this.coderConfigGroupBox.Text = "Конфігурація системи";
        //
        // redundancyGroupBox
        //
        this.redundancyGroupBox.Controls.Add(this.redundancyMacTrackBar);
        this.redundancyGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
        this.redundancyGroupBox.Location = new System.Drawing.Point(286,
21);

        this.redundancyGroupBox.Name = "redundancyGroupBox";
        this.redundancyGroupBox.Size = new System.Drawing.Size(264, 65);
        this.redundancyGroupBox.TabIndex = 4;
        this.redundancyGroupBox.TabStop = false;
        this.redundancyGroupBox.Text = "Надлишковість кодування";
        //
        // allVolCountGroupBox
        //
        this.allVolCountGroupBox.Controls.Add(this.allVolCountMacTrackBar);
        this.allVolCountGroupBox.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
        this.allVolCountGroupBox.Location = new System.Drawing.Point(12,
21);

        this.allVolCountGroupBox.Name = "allVolCountGroupBox";
        this.allVolCountGroupBox.Size = new System.Drawing.Size(264, 65);
        this.allVolCountGroupBox.TabIndex = 3;
        this.allVolCountGroupBox.TabStop = false;
        this.allVolCountGroupBox.Text = "Довжина коду";
        //
        // toolTip
        //
        this.toolTip.AutomaticDelay = 2000;
        this.toolTip.AutoPopDelay = 20000;
        this.toolTip.InitialDelay = 2000;
        this.toolTip.ReshowDelay = 1000;
        //
        // redundancyMacTrackBar
        //
        this.redundancyMacTrackBar.BackColor =
System.Drawing.Color.Transparent;
        this.redundancyMacTrackBar.BorderColor =
System.Drawing.SystemColors.ActiveBorder;
        this.redundancyMacTrackBar.Font = new System.Drawing.Font("Verdana",
8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte) 0));
        this.redundancyMacTrackBar.ForeColor =
System.Drawing.Color.FromArgb(((int) ((byte) (123)))), ((int) ((byte) (125))),
((int) ((byte) (123))));
        this.redundancyMacTrackBar.IndentHeight = 6;

```

```

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

```

```

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

```

```

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

```

```

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

```

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

    #endregion

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

Файл Code_Rid_Solomon_Decoder.cs - декодер коду Ріда-Соломона

```

using System;
using System.Threading;

namespace Data_Center
{
    /// <summary>
    /// Клас декодера коду Ріда-Соломона
    /// </summary>
    public class Code_Rid_Solomon_Decoder : Code_Rid_Solomon_Base
    {
        #region Data

        // Массив 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 Code_Rid_Solomon_Decoder()
        {
            // 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 Code_Rid_Solomon_Decoder(int dataCount, int eccCount, int[]
        volList)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, volList,
            (int)Code_Rid_Solomon_Type.Alternative);

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

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

            // Створюємо об'єкт класу роботи з елементами поля Галуа

```

```

        this.eGF16 = new GF16();
    }

#endregion Construction & Destruction

#region Public Operations

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

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

        } else
        {
            maxVolCount = (int)Code_Rid_Solomon_Const.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.eCode_Rid_Solomon_Type)
            )
            {
                this.mainConfigChanged = true;
            }

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

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

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

```

```

стадії
        // Кількість ітерацій, що відслідковуються прогресом, на першій
        // залежить від типу використовуваної матриці
        if (this.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.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
    {
        //...вказуємо на помилку конфігурації
        this.configIsOK = false;
    }

    return this.configIsOK;
}

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

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

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

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

            data[i] = mulSum;

```

```

        } 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
звотної
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
звотної
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)
        &&
        (OnUpdateCode_Rid_Solomon_MatrixFormingProgress != null)
    )
    {
        //...Виводимо дані
        OnUpdateCode_Rid_Solomon_MatrixFormingProgress((((double) (k
+ 1) / (double) this.n) * percOfSecondStage) + percOfFirstStage);
    }

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

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

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

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

        return false;
    }
}

return true;
}

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

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

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

/// <summary>

```

```

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

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

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

        return;
    }

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

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

    //...і вектор есс-томів для "затикання" пробілів, створених
    // загубленими основними томами
    int[] 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.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.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++)
{
    // Шукаємося за списком томів доти, поки не знайдемо том для
    // відновлення для затикання "дірки" (основні томи мають номера
    // менше this.n (при нумерації з нуля!))
    while (this.volList[j] < this.n)
    {
        j++;
    }

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

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

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

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

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

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

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

```

```

MakeDispersal()
    // "автоматично" на попередньому етапі обробки
    for (int j = 0; j < this.n; j++)
    {
        this.FLog[i_n + j] = this.D[bs + j];
    }

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

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

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

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

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

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

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

        return;
    }

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

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

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

        return;
    }

    // Обчислюємо логарифми елементів інвертованої матриці
    LogFCalc();

```

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

// 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 Code_Rid_Solomon_Encoder.cs - кодер коду Ріда-Соломона

```

using System;
using System.Threading;

namespace Data_Center
{
    /// <summary>
    /// Клас кодера коду Ріда-Соломона
    /// </summary>
    public class Code_Rid_Solomon_Encoder : Code_Rid_Solomon_Base
    {
        #region Construction & Destruction

        /// <summary>
        /// Конструктор кодера за замовчуванням
        /// </summary>
        public Code_Rid_Solomon_Encoder()
        {
            // 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 Code_Rid_Solomon_Encoder(int dataCount, int eccCount)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount,
                (int)Code_Rid_Solomon_Type.Alternative);

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

        /// <summary>
        /// Розширений конструктор кодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="codecType">Тип кодера коду Ріда-Соломона (по типу
        матриці)</param>
        public Code_Rid_Solomon_Encoder(int dataCount, int eccCount, int
        codecType)
        {
            // Установка конфігурації кодера
            SetConfig(dataCount, eccCount, codecType);

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

        #endregion Construction & Destruction

        #region Public Operations

        /// <summary>
        /// Установка конфігурації кодера
        /// </summary>
        /// <param name="dataCount">Кількість основних томів</param>
        /// <param name="eccCount">Кількість томів для відновлення</param>
        /// <param name="codecType">Тип кодера кодера коду Ріда-Соломона (по
        типу матриці)</param>
        /// <returns>Булевський прапор операції установки конфігурації</returns>

```

```

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

    // Установлюємо константи, що відповідають обраному режиму
    if (codecType == (int)Code_Rid_Solomon_Type.Dispersal)
    {
        maxVolCount = (int)Code_Rid_Solomon_Const.MaxVolCountDisp;
    } else
    {
        maxVolCount = (int)Code_Rid_Solomon_Const.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.eCode_Rid_Solomon_Type)
        )
        {
            this.mainConfigChanged = true;
        }

        // Зберігаємо конфігурацію
        this.n = dataCount;
        this.m = eccCount;
        this.eCode_Rid_Solomon_Type = 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.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.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
        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.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.Dispersal)
        {
            //...робимо формування дисперсної матриці "D"
            if (!MakeDispersalMatrix())
            {
                // Указуємо, що кодер зконфігуровано некоректно
                this.configIsOK = false;

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

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

```

```

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

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

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

        return;
    }
}

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

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

    // Залежно від типу кодера беремо дані з відповідного масиву
    if (this.eCode_Rid_Solomon_Type ==
(int)Code_Rid_Solomon_Type.Dispersal)
    {
        // Формування рядка в матриці кодування
        for (int j = 0; j < this.n; j++)
        {
            // У матрицю кодування поміщаємо логарифми ii
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 ii
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 (OnCode_Rid_Solomon_MatrixFormingFinish != null)
{
    //...повідомляємо, що екземпляр класу готовий до роботи
    OnCode_Rid_Solomon_MatrixFormingFinish();
}

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

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

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

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

#endregion Private Operations
}
}
}

```

```

namespace RecoveryData
{
    partial class AboutForm
    {
        /// <summary>
        /// Необхідні змінні розробника.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

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

        #region Windows Form Designer generated code

        /// <summary>
        /// Необхідний метод для підтримки розробника - не модифікується
        /// зміст цього метода використовується редактором коду.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.developersListBoxdevelopersListBox = new
System.Windows.Forms.ListBox();
            this.Code_Rid_Solomon_IconTimer = new
System.Windows.Forms.Timer(this.components);
            this.okButtonXP = new PinkieControls.ButtonXP();
            this.SuspendLayout();
            //
            // developersListBoxdevelopersListBox
            //
            this.developersListBoxdevelopersListBox.BackColor =
System.Drawing.SystemColors.Control;
            this.developersListBoxdevelopersListBox.BorderStyle =
System.Windows.Forms.BorderStyle.None;
            this.developersListBoxdevelopersListBox.FormattingEnabled = true;
            this.developersListBoxdevelopersListBox.Items.AddRange(new object[]
{
                "МАГІСТЕРСЬКА ДИПЛОМНА РОБОТА",
                "",
                "На тему:",
                "",
                "Дослідження та програмна реалізація системи віртуалізації додатків
у ОС Linux",
                "",
                "",
                "Керівник: Дреева Г.М. ",
                "",
                "Розробив: студент Серга Андрій Володимирович",
                "                гр. КІ-23М",
                "",
                "м. Кропивницький 2024"}));
            this.developersListBoxdevelopersListBox.Location = new
System.Drawing.Point(11, 6);
            this.developersListBoxdevelopersListBox.Name =
"developersListBoxdevelopersListBox";

```

```

        this.developersListBoxdevelopersListBox.SelectionMode =
System.Windows.Forms.SelectionMode.None;
        this.developersListBoxdevelopersListBox.Size = new
System.Drawing.Size(330, 182);
        this.developersListBoxdevelopersListBox.TabIndex = 0;
        this.developersListBoxdevelopersListBox.TabStop = false;
        this.developersListBoxdevelopersListBox.SelectedIndexChanged += new
System.EventHandler(this.developersListBoxdevelopersListBox_SelectedIndexChanged
);
        //
        // Code_Rid_Solomon_IconTimer
        //
        this.Code_Rid_Solomon_IconTimer.Interval = 40;
        this.Code_Rid_Solomon_IconTimer.Tick += new
System.EventHandler(this.Code_Rid_Solomon_IconTimer_Tick);
        //
        // okButtonXP
        //
        this.okButtonXP.BackColor =
System.Drawing.Color.FromArgb(((int)((byte)(0))), ((int)((byte)(236))),
((int)((byte)(233))), ((int)((byte)(216))));
        this.okButtonXP.DefaultScheme = true;
        this.okButtonXP.DialogResult =
System.Windows.Forms.DialogResult.None;
        this.okButtonXP.Hint = "";
        this.okButtonXP.Location = new System.Drawing.Point(266, 177);
        this.okButtonXP.Name = "okButtonXP";
        this.okButtonXP.Scheme = PinkieControls.ButtonXP.Schemes.Blue;
        this.okButtonXP.Size = new System.Drawing.Size(75, 23);
        this.okButtonXP.TabIndex = 0;
        this.okButtonXP.Text = "OK";
        this.okButtonXP.Click += new
System.EventHandler(this.okButtonXP_Click);
        //
        // AboutForm
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(350, 212);
        this.Controls.Add(this.okButtonXP);
        this.Controls.Add(this.developersListBoxdevelopersListBox);
        this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedDialog;
        this.MaximizeBox = false;
        this.MinimizeBox = false;
        this.Name = "AboutForm";
        this.ShowInTaskbar = false;
        this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterParent;
        this.Text = "Ипо попорпamy...";
        this.Load += new System.EventHandler(this.AboutForm_Load);
        this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.AboutForm_FormClosing);
        this.ResumeLayout(false);

    }

    #endregion

    private System.Windows.Forms.ListBox developersListBoxdevelopersListBox;
    private System.Windows.Forms.Timer Code_Rid_Solomon_IconTimer;
    private PinkieControls.ButtonXP okButtonXP;
}
}

```