

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

”Допущено до захисту”  
Завідувач кафедри кібербезпеки  
та програмного забезпечення  
д.т.н., професор  
\_\_\_\_\_ Олексій СМІРНОВ  
« \_\_\_\_ » \_\_\_\_\_ 2025 р.

**ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА**  
**за першим (бакалаврським) рівнем вищої освіти**  
на тему  
**“Програмне забезпечення системи динамічної хмарної**  
**платформи на основі сервісів IaaS”**

КБГЗ-2025

Виконав здобувач вищої освіти  
IV курсу, групи КІ-21-1  
ОПП «Комп’ютерна інженерія»  
спеціальності 123 «Комп’ютерна інженерія»  
\_\_\_\_\_ Гужявічус В.З.  
« \_\_\_\_ » \_\_\_\_\_ 2025 р.

Керівник проекту  
кандидат технічних наук, доцент  
\_\_\_\_\_ Лисенко І.А.  
« \_\_\_\_ » \_\_\_\_\_ 2025 р.  
Рецензент \_\_\_\_\_  
\_\_\_\_\_

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

ЗАТВЕРДЖУЮ

Завідувач кафедри

д.т.н., проф.

Олексій СМІРНОВ

« 17 » січня 2025 року

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

Гужявічусу Вітаутасу Зігмундасу

(прізвище, ім'я, по батькові)

1. Тема роботи Програмне забезпечення системи динамічної хмарної платформи на основі сервісів IaaS

2. Керівник роботи Лисенко Ірина Анатоліївна, канд. техн. наук

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

затверджені наказом вищого навчального закладу № 46-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту 23.05.2025 р.

4. Мета та завдання випускної кваліфікаційної роботи: Метою роботи є розробка програмного забезпечення системи динамічної хмарної платформи на основі сервісів IaaS

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

1. Призначення та область використання.

2. Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

Структурна схема системи 1 аркуш

Функціональна схема системи 1 аркуш

Діаграма процесів 1 аркуш

Блок-схема алгоритму роботи додатку 2 аркуша

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

### КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання  
« 17 » січня 2025 р.

Підпис керівника

Лисенко І.А.  
(прізвище та ініціали)

Завдання прийнято до виконання  
« 17 » січня 2025 р.

Підпис здобувача

Гужявічус В.З.  
(прізвище та ініціали)

## АНОТАЦІЯ

**Гужявічус В.З. Програмне забезпечення системи динамічної хмарної платформи на основі сервісів IaaS. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.**

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи динамічної хмарної платформи на основі сервісів IaaS.

Метою розробки є програмне забезпечення системи динамічної хмарної платформи на основі сервісів IaaS.

Результат роботи – програмна реалізація системи динамічної хмарної платформи на основі сервісів IaaS.

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

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

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

Програму розроблено в середовищі Visual C#.

**Ключові слова:** комп'ютерна інженерія, динамічна хмарна платформа, IaaS

## ABSTRACT

**Guzhyavičius V.Z. Software for a dynamic cloud platform system based on IaaS services. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.**

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a dynamic cloud platform system based on IaaS services.

The purpose of the development is software for a dynamic cloud platform system based on IaaS services.

The result of the work is a software implementation of a dynamic cloud platform system based on IaaS services.

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

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

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

The program was developed in the Visual C# environment.

**Keywords:** computer engineering, dynamic cloud platform, IaaS



## 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
IaaS	– Infrastructure-as-a-Service
LDPC	– коди Галлагера
NAK	– негативне підтвердження
PaaS	– Platform-as-a-Service
SaaS	– Software-as-a-Service

					ВКРБ-123.25.0067.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		2

## ВСТУП

**Актуальність теми.** Хмарні сервіси, насамперед послуги з надання віртуалізованих обчислювальних ресурсів і ємності зберігання по моделі IaaS, виходять на передній план.

Миттєвий доступ до інформації, ефективна взаємодія із клієнтами й співробітниками поза залежністю від їхнього місцезнаходження стають усе більше вагомими факторами досягнення успіху в бізнесі. Нові умови жадають від корпоративних ІТ застосування інноваційних підходів і впровадження передових технологій. Хмари, Великі дані, спільна робота й мобільність роблять можливими нові прибуткові бізнес-моделі й разом з тим породжують попит на досить більші потужності, гнучкість і динамічність ІТ. Розвитку цього напрямку активно сприяють п'ять важливих тенденцій.

Нинішні лідери ринку відрізняються від своїх конкурентів тим, що їхні послуги доступні клієнтам у будь-який час і в будь-якому місці. Рецепт їхнього успіху простий: навіть перебуваючи в шляху, клієнти можуть переглядати послуги або товари, порівнювати їх, замовляти або використовувати. Беручи участь в оцінці продуктів або сервісів і коментуючи їх у соціальних мережах, користувачі включаються в ланцюжок створення доданої вартості цих підприємств. Як показує сумний досвід ніколи успішних продавців товарів поштою й мережними магазинами із продажу косметичних засобів, якщо запізнитися з наданням клієнтам нових можливостей, можна швидко зійти з дистанції.

**Мета й завдання дослідження.** Метою роботи є програмне забезпечення системи динамічної хмарної платформи на основі сервісів IaaS.

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

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

- Огляд існуючих систем динамічної хмарної платформи на основі сервісів IaaS.
- Дослідження системи динамічної хмарної платформи на основі сервісів IaaS.
- Програмна реалізація системи динамічної хмарної платформи на основі сервісів IaaS.

**Практична цінність отриманих результатів** полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі динамічної хмарної платформи на основі сервісів IaaS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи динамічної хмарної платформи на основі сервісів IaaS, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ\_2025

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		<b>4</b>

# 1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

## 1.1 Призначення системи

Інфраструктура як послуга (IaaS) – це важлива та дедалі популярніша послуга хмарних обчислень. У 2025 році IaaS готова до значного зростання на ключових світових ринках. Нижче в нашому галузевому звіті розглядаються деякі ключові тенденції, проблеми та тенденції для інвесторів, а також висвітлюються ключові гравці та висхідні зірки, яких варто враховувати для вашої портфельної стратегії.

IaaS вважається одним із чотирьох компонентів хмарних обчислень, поряд із платформою як послугою (PaaS), програмним забезпеченням як послугою (SaaS) та безсерверними обчисленнями. Світовий ринок є значним. У 2024 році світова ринкова капіталізація становила 108,34 мільярда доларів. Це більше, ніж 94,85 мільярда доларів у 2023 році, що відповідає сукупному річному темпу зростання (CAGR) у 14,2%.

Очікується, що протягом наступних років зростання ще більше прискориться. До 2028 року ринок може досягти 212,34 мільярда доларів (18,3% CAGR). Ринок можна розділити на кілька сегментів. IaaS поділяється на багато вертикалей за галузями, включаючи ІТ та телекомунікації, охорону здоров'я, роздрібну торгівлю та електронну комерцію, бізнес-інфраструктуру, виробництво та освіту.

Очікується, що охорона здоров'я демонструватиме найвищий середньорічний темп зростання (CAGR), оскільки все більше постачальників послуг переносять свою ІТ-інфраструктуру в хмару. Північна Америка виділяється як найбільший регіон за ринковою капіталізацією IaaS (58,37 млрд доларів США). Однак Азіатсько-Тихоокеанський регіон відзначається найшвидшим впровадженням швидших інтернет-з'єднань (наприклад, 4G). У

					ВКРБ-123.25.0067.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

2025 році Південна Америка, Індія та Китай також очікують більшого зростання цифровізації.

IaaS пропонує величезну перевагу для бізнесу в усьому світі – доступ до збережених даних для будь-кого, хто має підключення до Інтернету. Багатьом фірмам більше не потрібно мати власні сховища даних на місці (з їхніми ризиками безпеки та високими пов'язаними з ними витратами). Зокрема, стартапи вважають рішення IaaS безцінними завдяки їх швидкому розгортанню та вимогам до масштабованості.

COVID-19 також залишив свою спадщину. Бізнес досі пам'ятає часи роботи з дому та необхідність підтримувати надійні системи та процеси для віддаленої роботи. Пандемія 2020 року призвела до того, що більше фірм почали впроваджувати хмарні обчислювальні рішення, такі як IaaS. Завдяки цим рішенням, які тепер «вбудовані» в їхню бізнес-архітектуру, бізнес став більш комфортним та залежним від цього способу зберігання, маркування та відстеження інформації.

2025 рік стане свідком деяких важливих подій у сфері IaaS. Багато фірм звертаються до штучного інтелекту (ШІ), щоб дослідити можливості хмарних технологій на базі ШІ. Тут ШІ – це не просто ще один сервіс, що «працює» на хмарній платформі. ШІ, навпаки, є рушійною силою своїх операцій, допомагаючи запускати розумніші процеси та автоматизацію. Очікується, що результати включатимуть зниження витрат для фірм, підвищення ефективності та вищі рівні продуктивності.

На противагу деяким із цих тенденцій, також є свідчення переходу фірм до гібридних (або мультихмарних) стратегій. Хоча компанії дуже приваблюють економія коштів, масштабованість та гнучкість IaaS, багато хто прагне уникнути ризиків, пов'язаних із прив'язкою до постачальника. Деякі прагнуть вирішити цю проблему, поєднуючи публічні хмарні послуги від різних постачальників, що допомагає їм досягти більшої гнучкості та резервування на ринку, що швидко розвивається.

					ВКРБ-123.25.0067.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

З огляду на те, що понад половина світової капіталізації IaaS припадає на Північну Америку, не дивно, що ключові гравці переважно з США. Серед них відомі імена, такі як AWS (Amazon Web Services), Microsoft Azure, Google Cloud та IBM. Однак, Азія також може похвалитися деякими помітними гравцями, такими як LayerStack (Гонконг), Gigenet (Гонконг) та Alibaba Cloud (Китай).

Серед цих великих гравців з'являються деякі захопливі стартапи в IaaS. Вони часто пропонують унікальні бізнес-моделі – або орієнтуються на певні сегменти клієнтів з унікальними потребами в хмарних обчисленнях – щоб заповнити занедбані сфери в просторі IaaS, до яких великим компаніям може бути важко адаптуватися. Цікавим прикладом є NexGen Cloud. Ця IaaS-фірма, що рекламується як «Airbnb хмарних обчислень», спирається на масштабовані графічні карти (замість великих центрів обробки даних на процесорах). Рішення на 100% працюють на відновлюваній енергії, і зараз компанія демонструє унікальний спосіб розгортання хмарних рішень на базі штучного інтелекту без сильного негативного впливу на навколишнє середовище.

IaaS, як і інші сфери хмарних обчислень, залишається схильною до проблем хакерства. Витоки даних можуть траплятися, особливо коли підприємства погано встановили програмне забезпечення. В IaaS постачальник хмарних послуг відповідає за безпеку інфраструктури (наприклад, фізичних центрів обробки даних). Клієнти несуть відповідальність за захист своїх даних, програм та засобів контролю доступу. Коли клієнти неправильно розуміють цей взаємозв'язок, це може призвести до вразливостей у кібербезпеці.

Головною можливістю IaaS, звичайно ж, є прогнозований середньорічний темп зростання (CAGR) протягом наступних років. Включаючи відповідні акції IaaS до свого портфеля, інвестори можуть підживлювати обчислювальну потужність, необхідну для штучного інтелекту, машинного навчання, аналітики великих даних, Інтернету речей та блокчейн-додатків. Крім того, постачальники IaaS працюють за моделями передплати або оплати за використанням, забезпечуючи передбачувані та регулярні потоки доходів. Рівень утримання

					ВКРБ-123.25.0067.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7





«Сусіди», що працюють на одному PaaS-провайдеру, можуть впливати на продуктивність один одного. У випадку IaaS ви можете використовувати інстанси більших розмірів, що найчастіше означає виділений доступ до фізичної машини, у якої ні «сусідів». Більше число компонентів PaaS-системи означає більше крапок ризику, де може знадобитися втручання.

Хоча більшість PaaS-платформ надають універсальний інтерфейс для керування додатками. Підкупує те, з якою легкістю ви можете дублювати цілі середовища. Ви можете запустити клону всієї своєї системи всього за пару хвилин. От тільки вартість деяких платформ, наприклад, Heroku, може швидко стати занадто великою (навіть по мірках IaaS).

Вийшло так, що в деяких сферах виграє PaaS, а в інші – IaaS. Така реальність. Якщо ви можете дозволити собі більший ступінь ризику, то PaaS пропонує гнучкі можливості по керуванню додатками, що дозволяють значно прискорити цей процес, у порівнянні з IaaS.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи динамічної хмарної платформи на основі сервісів IaaS, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		<b>10</b>

## 2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

### **Cloud Storage від Google**

Cloud Storage від Google – це платформа для зберігання даних, яка допомагає компаніям будь-якого розміру зберігати дані за допомогою кількох параметрів автоматичного резервування. Функції включають керування версіями об'єктів, резервне копіювання даних, геонадлишкове сховище, дозволи на рівні об'єктів і кошиків, а також журнал аудиту. Платформа постачається з можливістю доставки сервера сховища, пристроєм передачі даних, що спрощує передачу даних користувачами в автономному режимі. Співробітники можуть зберігати дані в різних класах сховищ, таких як стандартне, nearline, coldline та архівне сховище. Функція турбо-реплікації Cloud Storage дозволяє користувачам реплікувати записи в інший регіон. Завдяки функції блокування кошика фахівці можуть налаштувати політику зберігання даних, щоб визначити період зберігання об'єкта в кошику. Cloud Storage пропонує інтеграцію з різними сторонніми програмами, такими як Acronis, CloudBerry Lab, Comet, Cohesity та іншими. Ціноутворення залежить від обсягу та тривалості зберігання даних, а підтримка надається через документацію, телефон і чат.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		11

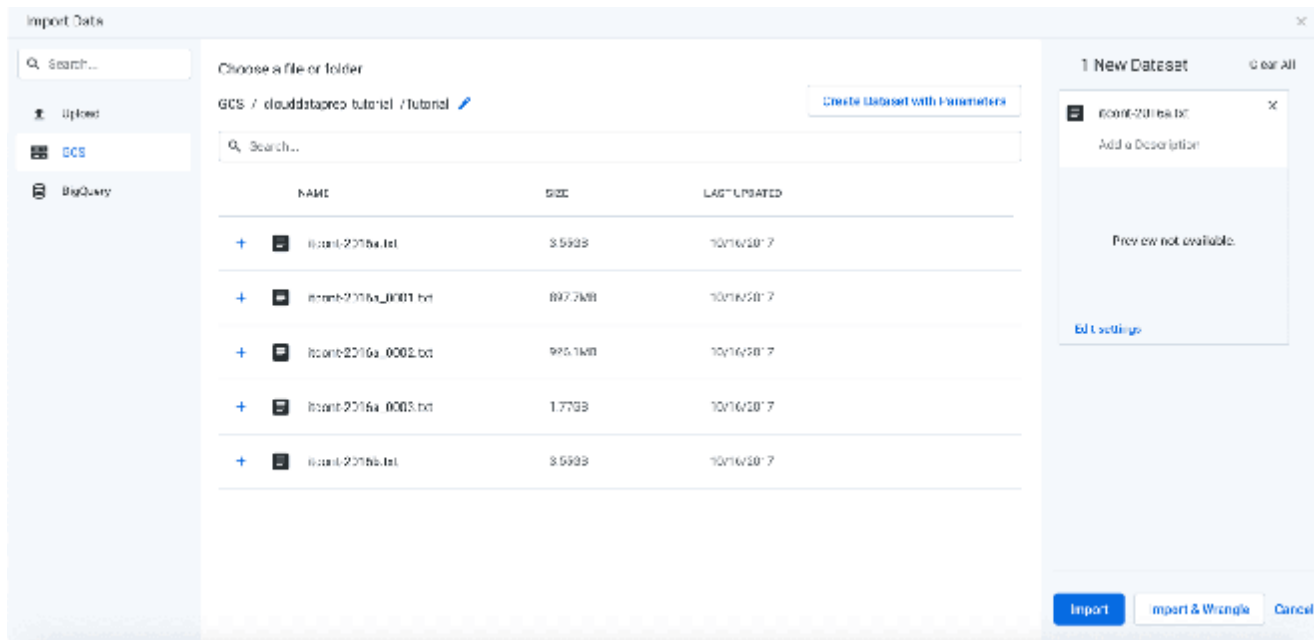


Рисунок 2.1 – Знімки екрана хмарного сховища Google

### Site24x7

Site24x7 – це хмарна платформа для моніторингу веб-сайтів і серверів, яка допомагає малому та великому бізнесу контролювати веб-сайти, сервери, хмари, мережі, програми та користувачів у режимі реального часу. Платформа дозволяє користувачам отримувати аналітику про час безвідмовної роботи, продуктивність веб-програм, інтернет-сервіси та приватні мережі. Основні функції Site24x7 включають моніторинг продуктивності веб-сайту, моніторинг швидкості веб-сторінок (браузера), DNS-сервери, доступність веб-сайту, моніторинг пошкодження веб-сайту, швидкість протоколу передачі файлів (FTP), API передачі репрезентативного стану (REST), сертифікати Secure Sockets Layer (SSL)/Transport Layer Security (TLS) та багато іншого. Site24x7 також забезпечує мобільний доступ. Продукт доступний за щомісячною підпискою, а підтримка надається телефоном, електронною поштою та через онлайн-службу підтримки.

### Google Cloud

Google Cloud – це набір хмарних обчислювальних сервісів, що дозволяє компаніям створювати, розгортати та масштабувати додатки. Платформа

обслуговує широкий спектр галузей, таких як роздрібна торгівля, фінансові послуги, охорона здоров'я, медіа, телекомунікації, ігри, виробництво, ланцюги поставок, уряд, освіта та автомобільна промисловість. В основі Google Cloud лежить технологія, за допомогою якої компанії можуть створювати, розгортати додатки та аналізувати дані. Платформа пропонує Gemini 2.0 та Google Agentspace. Це включає агентів ШІ, пошук на основі ШІ та NotebookLM для підприємств. Vertex AI – це повністю керована платформа ШІ, вдосконалена Gemini. Вона надає доступ до кількох базових моделей. Це дає організаціям змогу створювати та масштабувати генеративні додатки ШІ. Contact Center AI також надає віртуальних агентів та продукти розмовного ШІ, такі як Speech-to-Text, для покращення обслуговування клієнтів. Інфраструктура Google Cloud включає Compute Engine. Платформа включає Google Kubernetes Engine та Cloud Run для автоматичного розгортання, масштабування та керування контейнерами. Cloud SQL – це повністю керований сервіс баз даних для MySQL, PostgreSQL та SQL Server. AlloyDB для PostgreSQL дозволяє підприємствам масштабувати робочі навантаження та створювати генеративні додатки ШІ. Бізнес також може використовувати BigQuery для масштабної аналітики та Looker – платформу для додатків для обробки даних бізнес-аналітики та вбудованої аналітики. Завдяки G-Suite та GCP, Google Cloud пропонує набір рішень, що забезпечують безпечні варіанти зберігання, інтегровані продукти для аналізу даних та обчислювальні можливості. Завдяки платформі G-Suite користувачі можуть встановлювати командні чати та співпрацювати над проектами за допомогою інструментів продуктивності, таких як Google Docs, Hangouts, Calendar та Drive. Крім того, G-Suite надає можливості налаштування для облікових записів Gmail користувачів. Центри обробки даних GCP по всьому світу складаються з фізичних активів, включаючи комп'ютери, жорсткі диски та інші віртуальні машини, що допомагають оптимізувати розподіл ресурсів, що забезпечує резервування у разі будь-якого збою або зменшення затримки. Надаючи глобальні, регіональні та зональні ресурси, GCP вдалося залучити людей до безсерверного середовища, що

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		13

усунуло потребу в будь-якій інфраструктурі. AppEngine на GCP допомагає масштабувати систему для автоматичного забезпечення необхідних ресурсів.

### **Atlantic.Net Cloud Platform**

Хмарна платформа Atlantic.Net допомагає організаціям запускати та масштабувати свій бізнес за допомогою хмарних сервісів та інфраструктури. Ключові функції включають хмарні віртуальні сервери, безпечне блочне сховище, знімки, відповідність нормативним вимогам, панель керування, виділені хости, хмарне резервне копіювання та керовані сервіси. Безпечна панель керування АСР дозволяє командам створювати та керувати хмарними віртуальними серверами, ключами SSH, записами DNS, блочним сховищем, приватними та публічними IP-адресами тощо з однієї платформи. Рішення дозволяє менеджерам відновлювати сервери до моменту створення знімка, створювати нові сервери зі знімків або переносити копії знімків в інше місце.

### **Hevo**

Hevo – це двонаправлена платформа для конвеєрів даних без кодування, спеціально розроблена для сучасних потреб ETL, ELT та Reverse ETL. Вона допомагає командам обробки даних оптимізувати та автоматизувати потоки даних у масштабах всієї організації, що призводить до економії ~10 годин інженерного часу на тиждень та в 10 разів швидшої звітності, аналітики та прийняття рішень. Платформа підтримує понад 100 готових до використання інтеграцій з базами даних, SaaS-додатками, хмарними сховищами, SDK та потоковими сервісами. Понад 500 компаній, орієнтованих на дані, у понад 35 країнах довіряють Hevo для своїх потреб в інтеграції даних. Спробуйте Hevo сьогодні та запустіть свої повністю керовані конвеєри даних лише за кілька хвилин.

### **Oxylabs**

Oxylabs – провідна платформа для збору веб-аналітики та преміальний постачальник проксі-серверів, що допомагає компаніям будь-якого розміру використовувати великі дані для прийняття стратегічних рішень. Ми заслужили

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		14

довіру понад 1000 клієнтів по всьому світу, включаючи компанії зі списку Fortune Global 500, академічні установи та дослідників. Вони покладаються на OxyLabs для таких випадків використання, як дослідження ринку, перевірка реклами, SEO-моніторинг та захист бренду. Наша розгалужена мережа проксі-серверів, що охоплює понад 102 мільйони IP-адрес у 195 країнах, забезпечує безпрецедентне глобальне покриття. Наша команда підтримує цю надійну інфраструктуру, яка цілодобово контролює системи, щоб забезпечити високу надійність та стабільність. Наші індивідуальні рішення надають клієнтам ефективний доступ до необхідних їм даних, незалежно від їхньої галузі чи масштабу. Ідеальний користувач. Наші рішення ідеально підходять для бізнесу, який прагне збирати дані для зростання та інновацій. Незалежно від того, чи ви цифровий маркетолог, який відстежує ефективність SEO, аналітик даних, який контролює ціни конкурентів, чи фахівець з кібербезпеки, який захищає цілісність бренду, OxyLabs пропонує інструменти, необхідні для успіху. Ми підтримуємо як стартапи, так і підприємства, роблячи збір даних безперебійним та доступним. Унікальні переваги. Комплексні рішення для веб-скрейпінгу. Ми пропонуємо низку типів проксі-серверів, адаптованих до конкретних потреб: Проксі-сервери для центрів обробки даних: Високопродуктивні проксі-сервери для дослідження ринку, захисту бренду та безпеки електронної пошти. Проксі-сервери для житлових приміщень: Імітують поведінку реальних користувачів, ідеально підходять для перевірки реклами, тестування навантаження та моніторингу відгуків. Web Unblocker: Рішення на базі штучного інтелекту для вилучення даних без блокування зі складних веб-сайтів, ідеальне для парсингу нерухомості та агрегації тарифів на подорожі. Web Scraper API: Єдине потужне рішення для масштабованого вилучення даних з широкого спектру веб-сайтів. Воно надає точні та надійні дані з високим рівнем успішності, що робить його ідеальним для таких випадків використання, як дослідження ринку, моніторинг цін, аналіз настроїв та розвідка конкурентів. Глобальне охоплення Маючи понад 102 мільйони IP-адрес у 195 країнах, OxyLabs пропонує один з найбільших пулів

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		15

проксі-серверів, що забезпечує комплексний глобальний доступ до даних. Надійна підтримка та керівництво Наша команда експертів доступна цілодобово, щоб допомогти з технічними проблемами та максимізувати цінність наших послуг. Ми надаємо не лише інструменти, але й персоналізовану підтримку для досягнення ваших цілей збору даних. Галузі, які ми обслуговуємо Oхуlabs пропонує індивідуальні рішення для різних галузей: Маркетинг: проведення конкурентного аналізу, моніторинг реклами та аналіз настроїв клієнтів. Електронна комерція: моніторинг цін конкурентів, збір відгуків та оптимізація пропозицій ваших продуктів. Фінанси: аналіз ринкових тенденцій та підтримка фінансових досліджень. Кібербезпека: захист вашого бренду, моніторинг цифрових активів та запобігання шахрайству. Підтримка клієнтів Ми віримо в побудову міцних відносин з нашими клієнтами, пропонуючи постійну підтримку з першого дня. Наша спеціалізована команда надає допомогу з адаптацією, технічну підтримку та постійні консультації, щоб допомогти вам отримати максимальну віддачу від наших рішень. Незалежно від того, чи потрібна вам допомога з інтеграцією нашого Web Scraper API, чи з вибором правильного типу проксі-сервера для вашого випадку використання, ми готові підтримати вас на кожному кроці. Чому варто обрати Oхуlabs? Наша відданість інноваціям, прозорості та задоволенню потреб клієнтів відрізняє нас від інших. Ми розуміємо, що кожен бізнес має унікальні потреби, і ми тісно співпрацюємо з нашими клієнтами, щоб пропонувати рішення, які дають результати. Незалежно від того, чи ви малий бізнес, який прагне покращити ринкове розуміння, чи велике підприємство, яке шукає масштабовані можливості збору даних, Oхуlabs – ваш надійний партнер у використанні можливостей великих даних. Відчуйте різницю з Oхуlabs і розширте можливості свого бізнесу за допомогою надійних, ефективних та етичних рішень для збору даних, які відкривають нові можливості для зростання та успіху.

### **Amazon Lightsail**

Lightsail – це сервісне рішення «Інфраструктура як послуга» (IaaS),

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

розроблене для того, щоб допомогти малому бізнесу, студентам, розробникам та іншим фахівцям розміщувати програми, веб-сайти, програмне забезпечення та середовища для розробників або тестування в хмарі. Платформа автоматично налаштовує доступ, середовища безпеки, мережу або інші специфікації та включає статичну IP-адресу, моніторинг ресурсів та можливості керування системою доменних імен (DNS). Lightsail дозволяє компаніям створювати віртуальні сервери, запускаючи операційні системи, попередньо налаштовані програми або стек розробки, включаючи такі інструменти, як WordPress, Plesk, Lamp та інші. Розробники можуть отримувати сповіщення електронною поштою або текстовими повідомленнями про розбіжності та розгортати багаторівневі програми відповідно до вимог. Крім того, його можливості балансування навантаження спрямовують веб-трафік між віртуальними серверами та захищають їх від варіацій та збоїв. Lightsail пропонує інтерфейс прикладного програмування (API), який допомагає компаніям інтегрувати платформу з кількома сторонніми системами. Продукт доступний за щомісячною підпискою, а підтримка розширюється через поширені запитання, документацію, відеоуроки та інші онлайн-заходи.

## **2.2 Обґрунтування вибору засобів для побудови системи та мови програмування**

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

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		17

покликані спростити розробку додатків мовою Visual C# версії 5.0 і .NET Framework версії 4.5.

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

Як об'єктно-орієнтована мова, Visual C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод Main – крапку входу додатка – інкапсулюються у визначення класів. Клас може успадковувати безпосередньо з одного родового класу, але може реалізовувати будь-яке число інтерфейсів. Для методів, які перевизначають віртуальні методи в батьківському класі, необхідно ключове слово `override`, щоб виключити випадкове повторне визначення. У мові Visual C# структура схожа на полегшений клас: це тип, що розподіляється по стопках, що реалізує інтерфейси, але не підтримує спадкування.

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

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

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		18

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

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

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

### **Архітектура платформи .NET Framework**

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

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

містить маніфест із відомостями про типи складання, версії, мови й регіональні параметри та вимоги безпеки.

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

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

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		20

## 2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи динамічної хмарної платформи на основі сервісів IaaS.

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

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

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

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

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

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.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		21

## 3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

### 3.1 Опис функціонування системи

Панель керування віртуальною інфраструктурою як послугою (IaaS) розроблена на основі наступних принципів:

- Широкі функціональні можливості.
- Інтуїтивно зрозумілий інтерфейс.
- Повна автоматизація.

Всі дії в панелі керування здійснюються в кілька клічів.

#### Можливості панелі керування

##### Створення віртуальних серверів

Час створення сервера – 2 хвилини. Доступні різні шаблони (Windows Server, FreeBSD, CentOS, Ubuntu, Debian і ін.).

##### Зміна конфігурацій серверів

У будь-який момент ви можете додати або зменшити RAM і CPU, збільшити розмір диска, тим самим адаптуючи вашу інфраструктуру під поточні потреби. Зміни набудуть чинності після перезавантаження сервера. Зменшувати диск не можна. Якщо вам потрібно додати дисковий простір на час, краще підключити додатковий диск, що згодом можна видалити.

##### Зміна типу диска й додаткові диски.

Ви можете додавати до вашого віртуального сервера додаткові SAS, SSD або SATA диски.

Так само можлива міграція з SAS на SSD і назад.

Обидві операції відбуваються на лету й не вимагають перезавантаження сервера.

##### Приватні мережі

Створюйте приватні мережі й поєднуйте в них свої віртуальні сервера.

					ВКРБ-123.25.0067.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

Пропускна здатність каналу усередині приватної мережі 1Гбіт/с.

### **Створення власних шаблонів**

З раніше настроєного сервера ви можете створити власний шаблон, після чого розвертати з нього нові віртуальні машини з налаштуваннями операційної системи й програмного забезпечення, застосованими із цього шаблону. Це дозволить вам автоматизувати повторювані процедури розгортання нових серверів.

### **Снапшоти**

Снапшот (Snapshot) – миттєвий знімок стану сервера.

Створюйте снапшоти перед внесенням значимих змін на вашім віртуальному сервері. У випадку негативних наслідків ви можете швидко відкотитися на стан, зафіксований у снапшоті.

### **Керування резервним копіюванням**

Настройте резервне копіювання ваших віртуальних машин і не турбуйтеся за схоронність даних на них.

### **Керування DNS**

Делегуйте свої домени на наші NS-сервери й управляйте їхніми ресурсними записами безпосередньо з панелі керування.

### **Замовлення додаткових ліцензій**

У розділі "Налаштування-Ліцензії" на сторінці вашого сервера ви можете при необхідності замовити активацію додаткових ліцензій.

### **Поповнення балансу**

Система підтримує різні способи оплати через популярні платіжні системи: оплата картою, готівкою, з електронного гаманця й т.д.

### **Фінансові документи для юридичних осіб**

Безпосередньо в панелі керування ви можете переглянути й скачати акти виконаних робіт з виставленого на оплату рахунком. Укажіть окремий e-mail адреса для доставки фінансових документів, щоб ми відправляли їх вашому бухгалтерові на початку кожного місяця.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

## **Звернення до служби підтримки**

Ви можете створити звернення до нашої служби підтримки прямо в панелі керування й одержати потрібну вам консультацію 24 години на добу 365 днів у році. Докладніше про нашу службу підтримки читайте тут

## **Інформація з білінгу**

Переглядайте докладну інформацію з операцій з вашим особовим рахунком.

## **База знань**

У нашій базі знань зібрані покрокові користувальницькі інструкції з найпоширеніших питань роботи сервісу й адміністрування ПЗ, що виникає в наших клієнтів.

## **Статистика навантаження на встаткування**

У вкладці "Продуктивність" меню панелі керування серверами відображається актуальна інформація про навантаження на вашу віртуальну машину. Доступні графіки продуктивності за мінуту, день, місяць і рік.

## **Аналітика по партнерській програмі**

У розділі "Партнерам" веб-панелі доступна інформація про партнерські нарахування й про кількість притягнутих вами користувачів по реферальній програмі. Там же ви знайдете ваше персональне реферальне посилання.

## **Двофакторна аутентифікація**

Активуйте другий фактор аутентифікації (СМС) для забезпечення додаткового рівня безпеки вашого аккаунта.

## **Зміна пароля**

Ви можете в будь-який момент змінити пароль вашого облікового запису в розділі "Налаштування" панелі керування.

## **Активація промо-коду**

У вас є промо-код? Активуйте його в розділі "Налаштування" для одержання додаткових бонусів.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		24

### 3.2 Розробка структурної схеми

Ключовим фактором застосування інноваційних підходів є проривні технології, у яких є достатній потенціал довгострокового впливу з погляду вироблених змін у суспільстві, культурі й на ринках збуту. До їхнього числа ставляться хмарні обчислення, мобільність, спільна робота, Великі дані, а також широкополосна інфраструктура, що забезпечує гнучке й масштабоване надання ІТ-ресурсів (ємності зберігання, обчислювальних потужностей і додатків) у міру необхідності. Завдання ІТ-фахівців полягають у тому, щоб перетворити наявні в їхньому розпорядженні ІТ-системи й інфраструктури в динамічні рішення, що орієнтуються на поточні потреби.

#### Обчислювальні потужності із хмари

Відповідно до огляду хмарних рішень за 2024 рік (Cloud-Monitor 2024), підготовленому консалтинговою компанією KPMG, галузевою асоціацією Vitkom і аналітичною компанією PwC, усе більше організацій роблять вибір на користь віртуалізованих ІТ-інфраструктур. Так, уже 41% підприємств із числом співробітників більше 2000 чоловік одержує свої обчислювальні ресурси із хмари, причому тенденція до росту цього показника зберігається (KPMG і Vitkom, Cloud-Monitor 2024, 02/2024). Вони зробили вибір на користь динамічного підходу до ІТ-сервісів, коли немає потреби здобувати власне апаратне забезпечення, а потрібні ресурси в найкоротший термін адаптуються до поточних потреб у міру необхідності. Компаніям не доводиться містити інфраструктуру, розраховану на пікові навантаження, при цьому оплачується тільки те, що дійсно використовувалося. Темі IaaS у даній сфері приділяється все більша увага.

Сервіс-провайдери, працюючи разом з розроблювачами спеціалізованого програмного забезпечення, прагнуть полегшити підприємствам вхід у мир хмарних обчислень, а також постійно розширюють асортименти пропозицій IaaS. У цей час можна виділити п'ять основних тенденцій.

					ВКРБ-123.25.0067.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		25

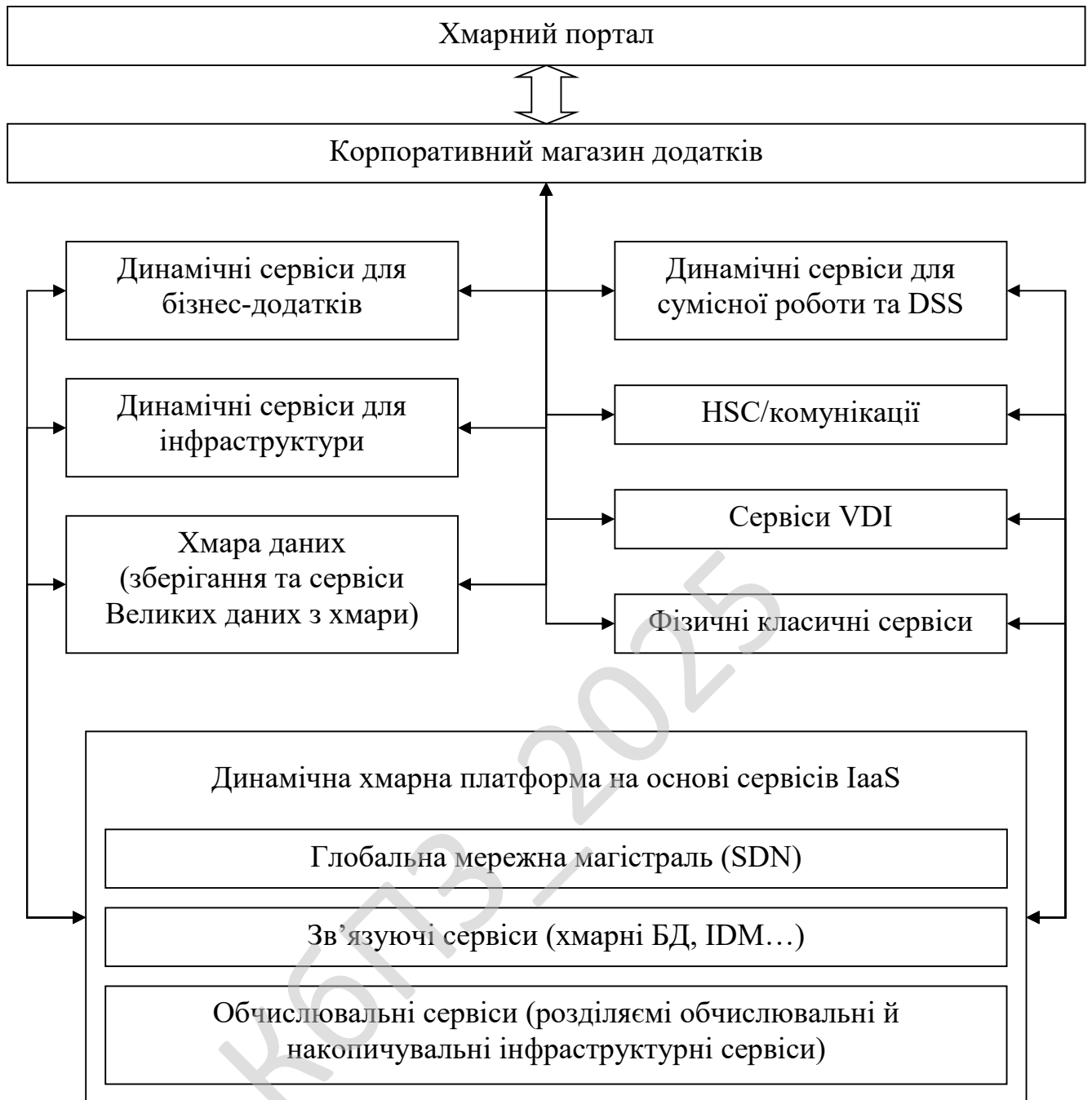


Рисунок 3.1 – Структурна схема системи

### Тенденція 1: гібридні хмари найбільш прості в реалізації

Для переважної більшості компаній гібридні хмари (Hybrid Cloud) є найпростішим способом підключення вже існуючих ІТ-середовищ до сервіс-провайдерів. Одна з головних передумов для цього – повноцінна взаємодія між внутрішніми й зовнішніми середовищами з погляду керування ІТ. Так, компанія

VMware, що лідирує на світовому ринку рішень для віртуалізації й хмарних інфраструктур, піклуючись про своїх клієнтів, уже сертифікувала десять провайдерів послуг IaaS як партнерів vCloud Datacenter Services. Ці провайдери одержують право на використання тих же самих моделей інфраструктури, керування й безпеки, які застосовуються самої VMware.

Одним натисканням клавіші підприємства можуть розширити свої віртуалізовані ІТ-ресурси за рахунок хмарних потужностей, надаваних партнерами vCloud Datacenter Services. Тим самим вони зможуть захистити інвестиції, зроблені в побудову інфраструктур для приватних хмар (Private Cloud), і не нести додаткових фіксованих витрат. Крім того, у результаті з'явилася можливість швидко й просто перемістити навантаження від одного сертифікованого партнера VMware до іншого.

Завдяки vCloud Connector від VMware, віртуальні машини легко переміщуються з локального середовища підприємства в середовище IaaS сервісу-провайдеру, сертифікованого як партнер VMware.

## **Тенденція 2: програмно обумовлені мережі**

«Концепція IaaS стає усе більше конкурентоспроможною й диференційованою, але в тому, що стосується реалізації мережного рівня, вона як і раніше перебуває на початку шляху», – констатувалося в торішньому аналізі тенденцій на 2024 рік, складеному аналітичним інститутом Ovum (Ovum, 2024 Trends to Watch: Private and Public Clouds, 11/2012). Лише кілька місяців через постачальники послуг IaaS одержали перший практичний досвід використання мережної технології, що, на думку експертів, здатна зробити революцію в цій галузі: згідно із прогнозами, до 2020 року програмно обумовленими ( Software-Defined Networking, SDN) стануть більше половини всіх мереж. Аналітики з IDC вважають, що світовий оберт на ринку продуктів SDN до 2024 року збільшиться з нинішніх приблизно 200 млн до 3,6 млрд доларів у рік (IDC, Worldwide Enterprise Networking 2024 Top 10 Predictions, 02/2024).

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		27

Концепція SDN базується на трьох основних ідеях. Перший крок – це віртуалізація мереж, тобто поділ фізичного й логічного рівнів, а також керування ними. Ще кілька років назад цей підхід почав активно застосовуватися відносно серверів. Другий крок – автоматизація більшої частини процесів мережного керування. У результаті користувачі рішень SDN одержують можливість повністю відокремити процеси експлуатації, адміністрування й налаштування мереж у своїх ландшафтах додатків від великих транспортних мереж (включаючи комутатори й маршрутизатори) і доручити керування ними різним командам. При цьому транспортні мережі створюють так званий нижележачий шар (Underlay), над яким провайдер послуг SDN може надбудувати для своїх клієнтів будь-яка кількість зовсім незалежних мережних областей (Overlay), включаючи виділені для конкретних замовників діапазони IP-адрес і мережні служби, такі як віртуальні брандмауери й балансувальники навантаження (Load Balancer).

Технологія SDN надає сервісам-провайдерам можливість упакування цілих ландшафтів додатків у настроєні відповідно до специфічних вимог віртуальні мережі, якими можна управляти як єдиним цілим. Клієнти одержують ці ландшафти у вигляді всеосяжного сервісу, що включає не тільки мережні служби, але також IT-ресурси (обчислювальні потужності, ємність зберігання) і компоненти додатків, тобто бази даних і сполучне програмне забезпечення (Middleware).

Відділення транспортної інфраструктури від мережних додатків надає мобільність цілим ландшафтам додатків і дозволяє переносити їх з одного ЦОД в інший без внесення змін у конфігурацію мережі. Приміром, якщо підприємство уклало договір на надання аутсорсингових послуг із зовнішнім сервісом-провайдером, перенос до нього всього виробничого середовища можна здійснити парою клацань мишею.

Завдяки технологіям SDN і IPv6, при необхідності можна переміщати цілі середовища додатків з одного хмарного ресурсу в інший.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		28

Концепція SDN дозволяє використовувати принципово нову модель хмарних сервісів, що забезпечує надзвичайно високу гнучкість, рухливість і економічність. Хоча на даний момент впровадження SDN ще тільки починається, очікується, що приблизно через два роки провайдери послуг IaaS зможуть реалізувати перші великомасштабні проекти. Тим часом великі постачальники мережного встаткування вже представили перші продукти з підтримкою технології SDN. Головна рушійна сила в цій сфері – некомерційне об'єднання Open Networking Foundation (ONF). Воно було засновано в 2011 році шістьома провідними мережними провайдерами, розроблювачами ПЗ й інтернет-компаніями; у цей час у ньому беруть участь уже більше 80 організацій.

### **Тенденція 3: Великі дані як сервіс**

Так звані Великі дані (Big Data) надають безлічі підприємств рідкий шанс впровадити нові моделі бізнесу й одержати додаткову додану вартість. Але фахівців у цій області небагато, а виходить, залучити їх до себе в штат непросто й відповідні послуги обходяться дорого. Багато підприємств, яким потрібно лише кілька разів на місяць проводити необхідний аналіз, шукають можливості одержання Больших Даних із хмар, щоб оплачувати тільки затребувані ними послуги. Можливість віртуалізації кластерів Hadoop забезпечує більші переваги для навантажень із високим ступенем рухливості (приміром, середовищ для перевірки концепцій, тестування й розробок, а також при пікових навантаженнях) і дозволяє перенести їх у середовище IaaS.

Основне завдання для постачальників послуги Big Data as a Service – це здатність надавати в одному середовищі послуги для декількох клієнтів одночасно (функція множинної оренди, MultiTenancy), з дотриманням всіх вимог по забезпеченню безпеки й ізольованості кожної із середовищ. Одночасно необхідно забезпечити гнучкість віртуалізації. Разом із сервісами-провайдерами над цим завданням зараз працюють такі великі постачальники рішень Hadoop, як Hortonworks і Cloudera. Доти поки ця концепція не стане зрілою, безпека у світі Hadoop і Больших Даних забезпечується шляхом поділу клієнтських середовищ.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		29



ще в 1981 році й здатний підтримувати не більше 4,3 млрд IP-адрес, досяг межі своїх можливостей уже досить давно, тому багато підприємств використовують усередині зони за маршрутизатором NAT ті самі частки IP-адреси, допускаючи їхнє накладення.

В аутсорсингових проектах, коли корпоративні мережі потрібно об'єднати із провайдерськими, це може викликати додаткові працеватрати, оскільки повторювані адреси прийде міняти. Протокол IPv6 розширює діапазон припустимих IP-адрес практично нескінченно й у такий спосіб дозволяє уникнути конфлікту адрес в аутсорсингових проектах.

### 3.3 Розробка функціональної схеми

Для стабільної роботи динамічної хмарної платформи на основі сервісів IaaS у даній роботі пропонується використовувати перешкодостійке кодування. На рисунку 3.2 зображена функціональна схема системи. З неї бачимо, що розроблена система підвищення стійкості до уражень інформації яка записана на динамічній хмарній платформі складається з наступних основних функціональних блоків:

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

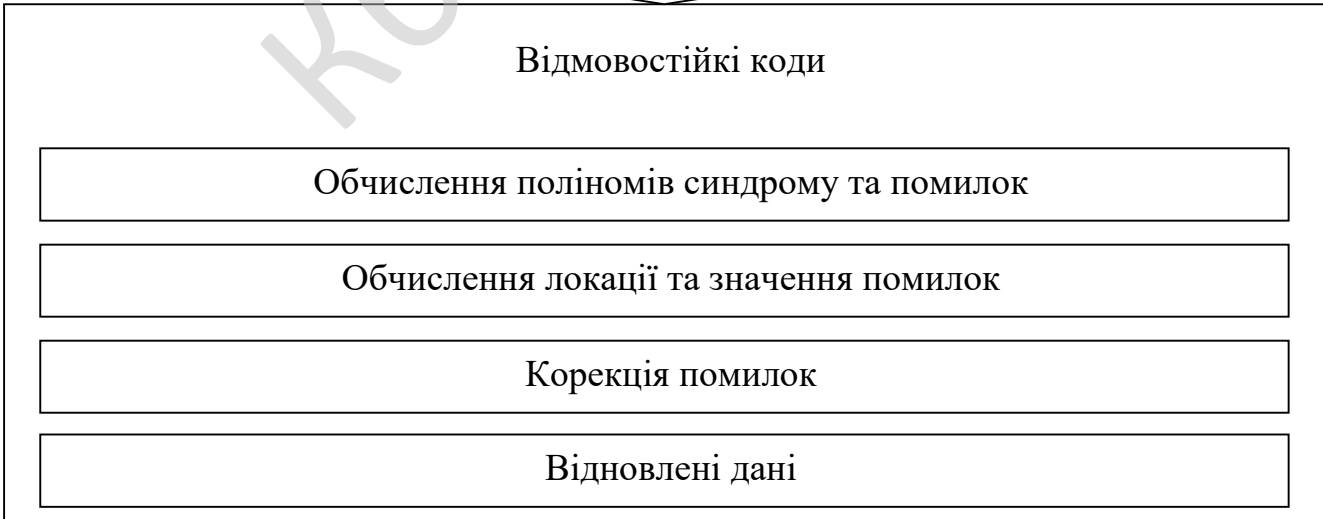
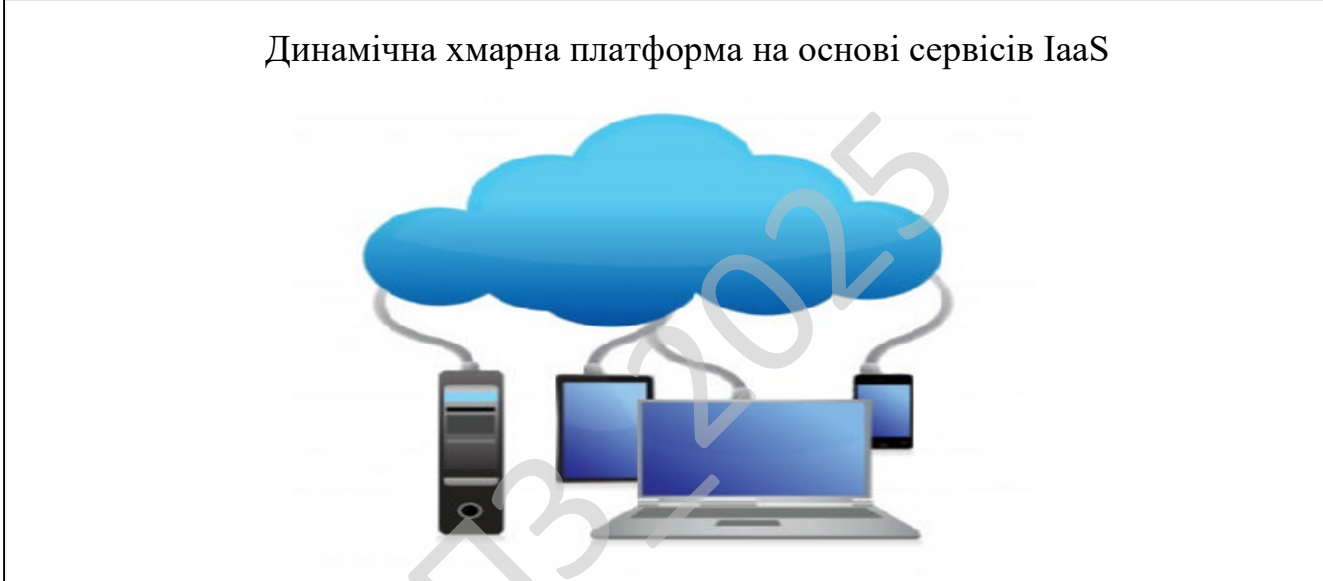
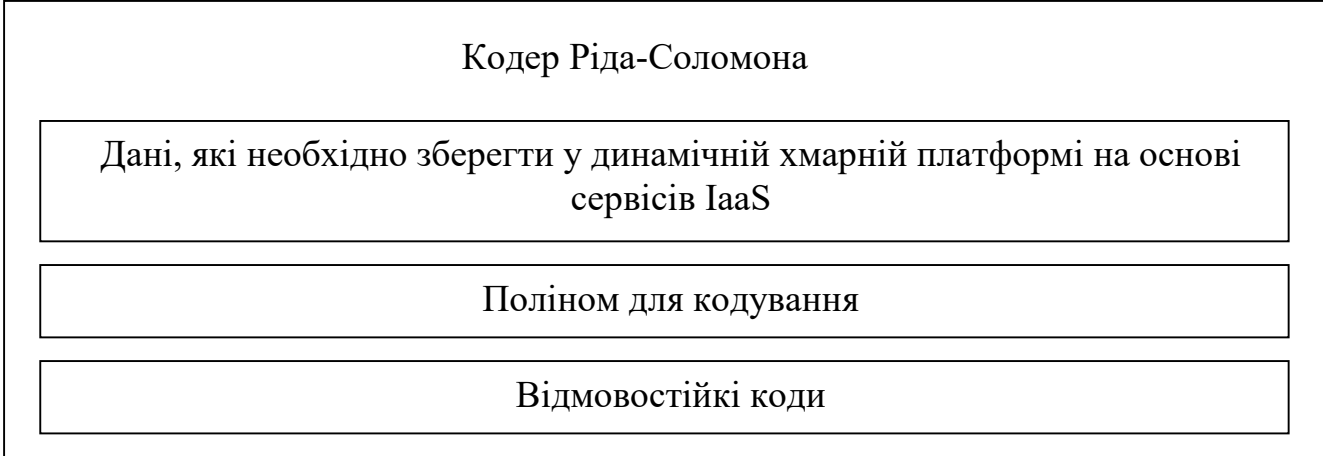


Рисунок 3.2 – Функціональна схема системи









Рисунок 3.3 – Діаграма взаємодії процесів

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

## 4 РЕАЛІЗАЦІЯ 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 динамічної хмарної платформи на основі сервісів IaaS.

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.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		37

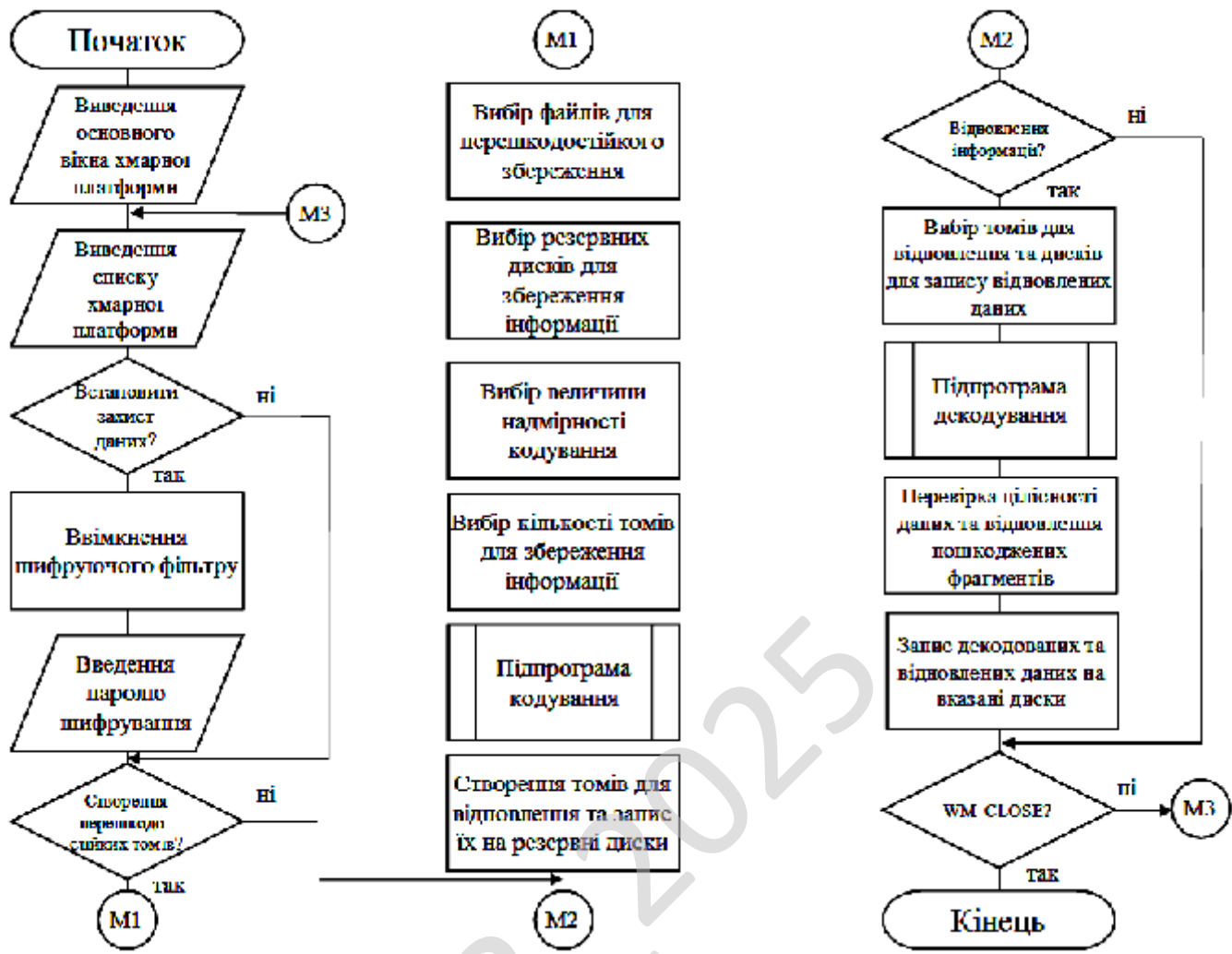


Рисунок 4.1 – Блок-схема основної програми

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.

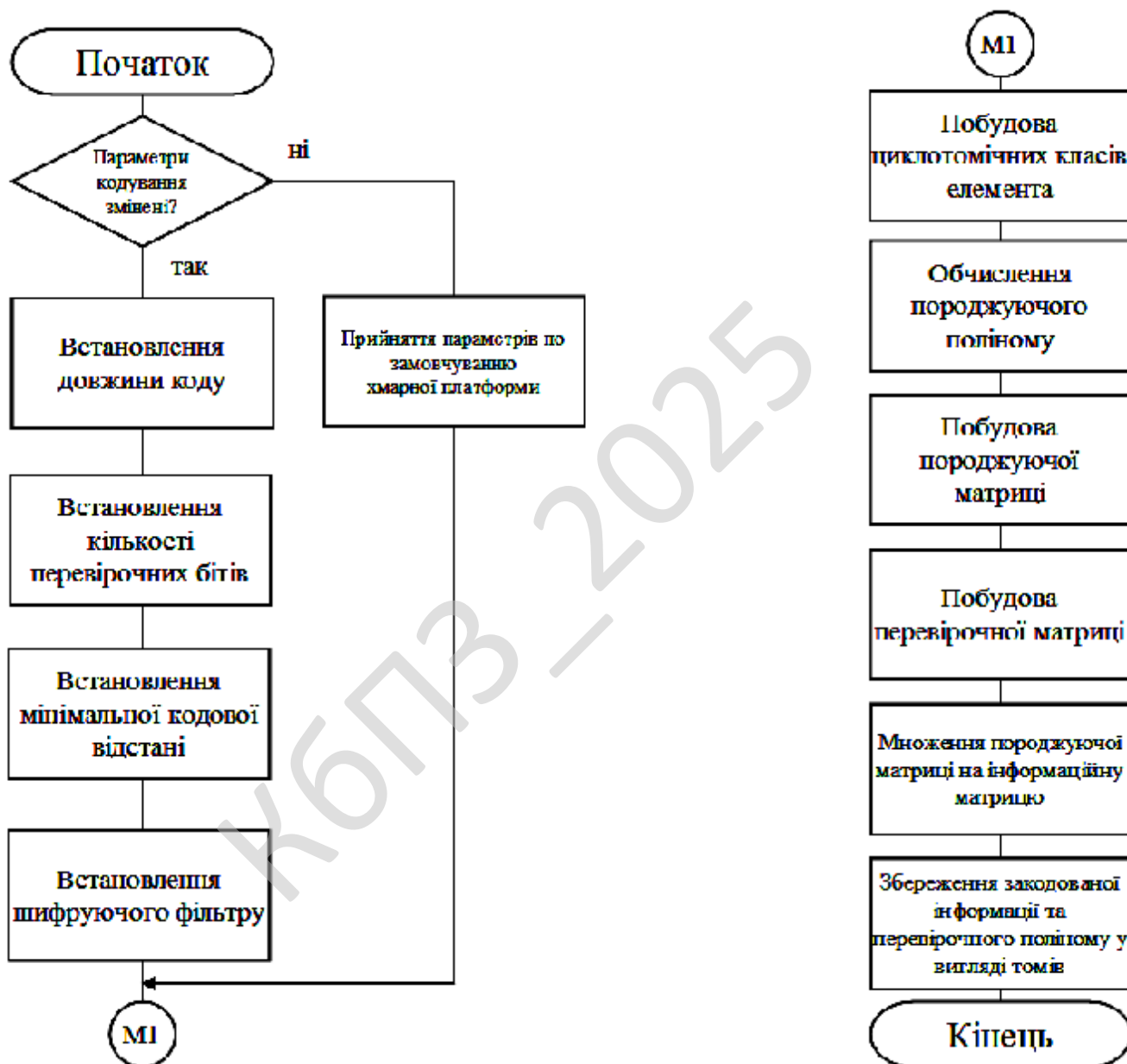


Рисунок 4.2 – Блок-схема роботи підпрограми

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

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

- Ведення декількох проектів.
- Гнучка система доступу з використанням ролей.
- Система відстеження помилок.
- Діаграми Ганта та календар.
- Ведення новин проекту, документів та управління файлами.
- Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.
- Власна Wiki для кожного проекту.
- Форуми для кожного проекту.
- Облік часових витрат.
- Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.
- Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Vazaar и Darcs).
- Створення записів про помилки на основі отриманих листів
- Підтримка LDAP автентифікації.
- Можливість самореєстрації нових користувачів.
- Багатомовний інтерфейс (у тому числі українська мова).
- Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (Gantt chart, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40

робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання.

На деяких діаграмах Ганта також показується залежність між завданнями. Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		41

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

- номер (ідентифікатор) дефекту;
- хто повідомив про дефект;
- дата і час виявлення дефекту;
- версія продукту, в якій виявлено дефект;
- серйозність (критичність) дефекту та пріоритет рішення;
- опис кроків для відтворення дефекту (неправильної поведінки програми);
- відповідальний за усунення дефекту;
- обговорення можливих рішень та їх наслідків;
- поточний стан виправлення дефекту;
- версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дампи пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		42



```

// ініціалізація поля біт парності нулями
    for (i = 0; i < n - k; i++) b[i] = 0;
// обробляємо всі символи
// вихідних даних праворуч ліворуч
    for (i = k - 1; i >= 0; i--i--)
        {
            feedback = index_of[data[i] ^ b[n - k - 1]];
// є ще символи для обробки?
            if (feedback != -1)
                {
// здійснюємо зрушення ланцюга vx-регістрів
                    for (j = n - k - 1; j > 0; j--j--)
// якщо поточний коефіцієнт g - це дійсний
// (тобто ненульовий коефіцієнт, те
// множимо feedback на відповідний g-коефіцієнт
// і складаємо його з наступних елементів ланцюжка
if (g[j] != -1) b[j] = b[j - 1] ^ alpha_to[(g[j] + feedback) % n];
else
        b[j] = b[j - 1];
        b[0] = alpha_to[(g[0] + feedback) % n];
            }
            else
                {
for (j = n - k - 1; j > 0; j--j--) b[j] = b[j - 1]; b[0] = 0;
}}}

```

**Нижче наведемо вихідний текст підпрограми декодера Ріда-Соломона.**

```

decode_rsolomon()
{
    int i, j, u, q;
    int s[n - k + 1]; // поліном синдрому помилки
    int elp[n - k + 2][n - k]; // поліном локатора помилки лямда
    int d[n - k + 2];
    int l[n - k + 2];
    int u_lu[n - k + 2],
    int count = 0, syn_error = 0,
    root[t], loc[t], z[t + 1], err[n], reg[t + 1];
// переводимо отримане кодове слово в індексну форму
// для спрощення обчислень
    for (i = 0; i < n; i++) recd[i] = index_of[recd[i]];
// обчислюємо синдром
//-----

```







```

if ((s[j] != -1) && (elp[u][i - j] != -1))
z[i] ^= alpha_to[(elp[u][i - j] + s[j]) % n];
// переводимо z[i] в індексну форму
z[i] = index_of[z[i]];
}
// обчислення значення помилок у позиціях loc[i]
for (i = 0; i < n; i++)
{
err[i] = 0;
// переводимо recd[] у поліноміальну форму
if (recd[i] != -1) recd[i] = alpha_to[recd[i]]; else recd[i] = 0;
}
// спочатку обчислюємо чисельник помилки
for (i = 0; i < l[u]; i++)
{
err[loc[i]] = 1;
for (j = 1; j <= l[u]; j++)
if (z[j] != -1)
err[loc[i]] ^= alpha_to[(z[j] + j * root[i]) % n];
if (err[loc[i]] != 0)
{
err[loc[i]] = index_of[err[loc[i]]];
q = 0 ; // формуємо знаменник коефіцієнта помилки
for (j = 0; j < l[u]; j++)
if (j != i)
q += index_of[1 ^ alpha_to[(loc[j] + root[i]) % n]];
q = q % n; err[loc[i]] = alpha_to[(err[loc[i]] - q + n) % n];
// recd[i] повинен бути в поліноміальній формі
recd[loc[i]] ^= err[loc[i]];
}
}
}
else // немає корінь,
// рішення системи рівнянь неможливо, тому що ступінь elp >= t
{
// переводимо recd[] у поліноміальну форму
for (i = 0; i < n; i++)
if (recd[i] != -1) recd[i] = alpha_to[recd[i]];
else
recd[i] = 0; // виводимо інформаційне слово як e
}
else // ступінь elp > t, рішення неможливо

```

```

{
// переводимо recd[] у поліноміальну форму
for (i = 0; i < n; i++)
if (recd[i] != -1)
recd[i] = alpha_to[recd[i]];
else
recd[i] = 0 ; // виводимо інформаційне слово як є
}
else // помилок не виявлено
for (i = 0; i < n; i++) if (recd[i] != -1) recd[i] = alpha_to[recd[i]];
else recd[i] = 0;
}

```

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile.

### **Гнучка́ розробка програмного забезпечення**

Agile software development, agile-методи – клас методологій розробки програмного забезпечення, що базується на ітеративній розробці, в якій вимоги та розв'язки еволюціонують через співпрацю між самоорганізовуваними багатофункціональними командами.

Гнучка розробка – найкращий засіб для підвищення продуктивності розробників програмного забезпечення.

Більшість гнучких методологій націлені на мінімізацію ризиків, шляхом зведення розробки до серії коротких циклів, що мають назву ітерацій, які зазвичай тривають один-два тижні. Кожна ітерація сама по собі виглядає як програмний проект в мініатюрі, і включає всі завдання, необхідні для видачі мінімального приросту за функціональністю: планування, аналіз вимог, проектування, кодування, тестування і документування. Хоча окрема ітерація, як правило, недостатня для випуску нової версії продукту, мається на увазі те, що гнучкий програмний проект готовий до випуску наприкінці кожної ітерації. Після закінчення кожної ітерації, команда виконує переоцінку пріоритетів розробки.

Agile акцентує увагу на безпосередньому спілкуванні «віч-на-віч». Більшість agile команд розташовані в одному офісі, його іноді називають bullpen. Як мінімум вона включає і «замовників» (замовники, які визначають продукт,

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		49

також це можуть бути менеджери продукту, бізнес аналітики або клієнти). Офіс може також включати тестувальників, дизайнерів інтерфейсу, технічних авторів і менеджерів.

Основною метрикою agile методів є робочий продукт. Віддаючи перевагу безпосередньому спілкуванню, agile-методи зменшують обсяг письмової документації в порівнянні з іншими методами. Це привело до критики цих методів як недисциплінованих.

Agile – родина процесів розробки, а не єдиний підхід в розробці програмного забезпечення, і визначається Agile Manifesto. Agile не включає практик, а визначає цінності та принципи, якими керуються успішні команди.

Agile Manifesto розроблений і прийнятий 17 розробниками 11-13 лютого 2001 року на лижному курорті The Lodge at Snowbird в горах Юти. Маніфест підписали представники наступних методологій Extreme programming, Scrum, DSDM, Adaptive software development, Crystal Clear, Feature driven development, Pragmatic Programming. Agile Manifesto містить 4 основні ідеї та 12 принципів. Примітно, що Agile Manifesto не містить практичних порад.

Основні ідеї:

- Особистості та їхні взаємодії важливіші, ніж процеси та інструменти;
- Робоче програмне забезпечення важливіше, ніж повна документація;
- Співпраця із замовником важливіша, ніж контрактні зобов'язання;
- Реакція на зміни важливіша, ніж дотримання плану.

Принципи, які роз'яснює Agile Manifesto:

- задоволення клієнта за рахунок ранньої та безперебійної поставки коштовного програмного забезпечення;
- вітання змін вимог навіть наприкінці розробки (це може підвищити конкурентоспроможність отриманого продукту);
- часта поставка робочого програмного забезпечення (кожен місяць або тиждень або ще частіше);

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		50

- тісне, щоденне спілкування замовника з розробниками впродовж всього проекту;
- проектом займаються мотивовані особистості, які забезпечені потрібними умовами роботи, підтримкою і довірою;
- рекомендований метод передачі інформації – особиста розмова (віч-на-віч);
- робоче програмне забезпечення – найкращий вимірник прогресу;
- спонсори, розробники та користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;
- постійну увагу поліпшенню технічної майстерності та зручному дизайну;
- простота – мистецтво не робити зайвої роботи;
- найкращі технічні вимоги, дизайн та архітектура виходять у самоорганізованій команді;
- постійна адаптація до мінливих обставин.

Маніфест та Принципи гнучкої розробки містять високорівневі ідеї щодо того, як потрібно вибудовувати процес розробки програмного забезпечення, щоб успішно завершувати проекти й створювати команди, в яких приємно та цікаво працювати.

Документи визначають, що потрібно для цього зробити, але не говорять, як це зробити. По-іншому й не могло бути, оскільки Маніфест та Принципи народилися внаслідок консенсусу представників різних (хоча й споріднених) напрямів, які могли знайти спільну основу лише на рівні базових цінностей та принципів.

Критика. Багато керівників проектів, що працюють у традиційних методологіях на кшталт «водоспаду», критикують agile-методи.

Один з повторюваних пунктів критики: при agile-підході часто нехтують створенням «дорожньої карти» розвитку продукту, так само як і управлінням вимогами, в процесі якого і формується така «карта».

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		51

Гнучкий підхід до управління вимогами не має на увазі далекосяжних планів (по суті, управління вимогами просто не існує в даній методології), а має на увазі можливість замовника раптом і несподівано наприкінці кожної ітерації виставляти нові вимоги, що часто суперечать архітектурі вже створеного і поставленого продукту. Таке іноді призводить до катастрофічних «авралів» з масовим рефакторингом і переробками практично на кожній черговій ітерації.

Крім того вважається, що робота в agile мотивує розробників вирішувати всі прибулі завдання найпростішим і найшвидшим можливим способом, при цьому часто не звертаючи уваги на коректність коду з точки зору вимог базової платформи (підхід «працює, та й добре», при цьому не враховується, що може перестати працювати при найменшій зміні або ж породити важкі до відтворення дефекти після реального розгортання у клієнта). Це призводить до зниження якості продукту і накопиченню дефектів.

Методології. Існують методології, які дотримуються цінностей і принципів заявлених в Agile Manifesto, деякі з них:

1. Agile Modeling – набір понять, принципів і прийомів (практик), що дозволяють швидко і просто виконувати моделювання і документування в проектах розробки програмного забезпечення. Не включає в себе детальну інструкцію з проектування, не містить описів, як будувати діаграми на UML.

Основна мета – ефективне моделювання і документування; але не охоплює програмування та тестування, не включає питання управління проектом, розгортання і супроводу системи. Однак включає в себе перевірку моделі кодом.

2. Agile Unified Process (AUP) спрощена версія IBM Rational Unified Process (RUP), розроблена Скоттом Амблером, яка описує просте і зрозуміле наближення (модель) для створення програмного забезпечення для бізнес-додатків.

3 Agile Data Method – група ітеративних методів розробки програмного забезпечення, в яких вимоги та рішення досягаються в рамках співпраці різних крос-функціональних команд.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		52

4. DSDM заснований на концепції швидкої розробки додатків (Rapid Application Development, RAD). Являє собою ітеративний і інкрементний підхід, який надає особливого значення тривалій участі в процесі користувача/споживача.

5. Essential Unified Process (EssUP).

6. Екстремальне програмування (Extreme programming, XP).

7. Feature driven development (FDD) – функціонально-орієнтована розробка.

Використовуване в FDD поняття функції або властивості (feature) Системи досить близько до поняття прецеденту використання, використовуваному в RUP, істотна відмінність – це додаткове обмеження: «кожна функція повинна допускати реалізацію не більше, ніж за два тижні». Тобто якщо сценарій використання досить малий, його можна вважати функцією. Якщо ж великий, то його треба розбити на декілька відносно незалежних функцій.

8. Getting Real – ітераційний підхід без функціональних специфікацій, що використовується для веб-додатків. У даному методі спершу розробляється інтерфейс програми, а потім її функціональна частина.

9. OpenUP – це ітераційно-інкрементний метод розробки програмного забезпечення. Позиціюється, як легкий і гнучкий варіант RUP. OpenUP ділить життєвий цикл проекту на чотири фази: початкова фаза, фази уточнення, конструювання та передачі. Життєвий цикл проекту забезпечує надання зацікавленим особам та членам колективу точок ознайомлення і прийняття рішень впродовж усього проекту. Це дозволяє ефективно контролювати ситуацію і вчасно приймати рішення про задовільність результатів. План проекту визначає життєвий цикл, а кінцевим результатом є остаточний додаток.

10. Scrum встановлює правила керування процесом розробки та дозволяє використовувати вже існуючі практики кодування, коректуючи вимоги або вносячи тактичні зміни. Використання цієї методології дає можливість виявляти і усувати відхилення від бажаного результату на більш ранніх етапах розробки програмного продукту.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		53

11. Бережлива розробка програмного забезпечення (lean software development). Використовує підходи з концепції бережливого виробництва.

#### 4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-2, який шифрує дані 256-бітними блоками з використанням 512-бітного ключа.

Допускається використання ключів менших розмірів (не менш 128 біт), які доповнюються бітовими нулями до 512 біт.

Шифруємий блок даних ділиться на 8 фрагментів по 32 біта (які позначені буквами  $A...H$ ).

Алгоритм виконує 64 раунду перетворень, у кожному з яких дані фрагменти обробляються в такий спосіб:

$$T = H_i + S_1(E_i) + Ch(E_i, F_i, G_i) + M_i + K_i,$$

$$H_{i+1} = G_i,$$

$$G_{i+1} = F_i,$$

$$F_{i+1} = E_i,$$

$$E_{i+1} = D_i + T,$$

$$D_{i+1} = C_i,$$

$$C_{i+1} = B_i,$$

$$B_{i+1} = A_i,$$

$$A_{i+1} = T + S_0(A_i) + Maj(A_i, B_i, C_i).$$

де  $T$  – тимчасова змінна.

Використовувані функції визначені в такий спосіб:

$$S_0(x) = (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22),$$

$$S_1(x) = (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25),$$

$$Ch(x, y, z) = (x \& y) \oplus (x' \& z),$$

$$Maj(x, y, z) = (x \& y) \oplus (x \& z) \oplus (y \& z),$$



де  $\gg$  – операція побітового зрушення (не циклічного) вправо.

Раунди розшифрування алгоритму виконуються у зворотній послідовності:

$$T = A_{i+1} + S_0'(B_{i+1}) + Maj'(B_{i+1}, C_{i+1}, D_{i+1}) + 2,$$

$$H_i = T + S_1'(F_{i+1}) + Ch'(F_{i+1}, G_{i+1}, H_{i+1}) + M_i' + K_i' + 4,$$

$$G_i = H_{i+1},$$

$$F_i = G_{i+1},$$

$$E_i = F_{i+1},$$

$$D_i = E_{i+1} + T' + 1,$$

$$C_i = D_{i+1},$$

$$B_i = C_{i+1},$$

$$A_i = B_{i+1}.$$

КБПЗ – 2025

					ВКРБ-123.25.0067.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		56



Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows 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.

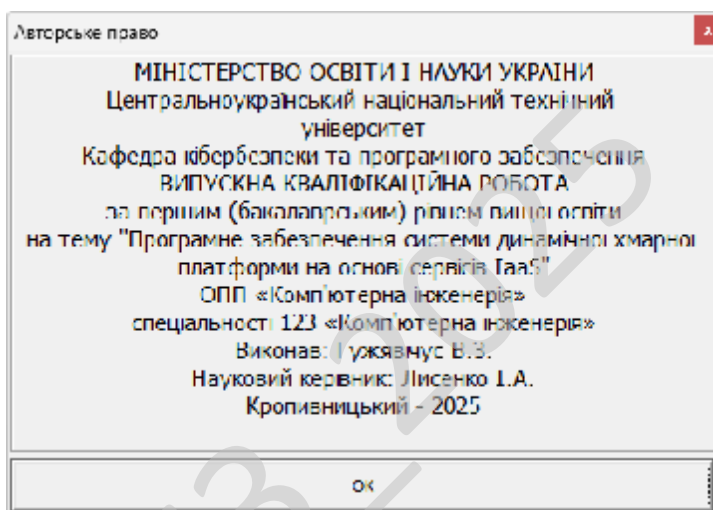


Рисунок 5.2 – Авторське право

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		58

## 6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи динамічної хмарної платформи на основі сервісів IaaS.

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

Рішення завдання полягало у вирішенні наступних задач:

- Був проведений огляд існуючих систем динамічної хмарної платформи на основі сервісів IaaS.
- Досліджена система динамічної хмарної платформи на основі сервісів IaaS.
- На основі отриманих результатів досліджень створена програмна реалізація системи динамічної хмарної платформи на основі сервісів IaaS.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання динамічної хмарної платформи на основі сервісів IaaS.

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

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

Програма реалізована на мові високого рівня Visual C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи динамічної хмарної платформи на основі сервісів IaaS. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		59

загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-2.

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

КБПЗ\_2025

					ВКРБ-123.25.0067.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		60

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.
2. Ramon Nastase «Computer Networking: The Beginner’s guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.
3. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.
4. Смірнова Т.В., Коноплицька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка*. 2024. №4(24), С. 6-27.
5. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології*, 2024, № 13, с. 28-35.
6. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.
7. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.
8. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchев, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings*, Volume 3530, 2023, pp. 256-265.
9. 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.

10. 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.

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

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

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

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

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

16. 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.

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

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

19. 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.

20. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) *Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies*, vol 48. Springer, Cham. 2021. pp 557-587.

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

22. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

23. 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.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		<b>63</b>

24. 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.

25. 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.

26. 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.

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

28. 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.

29. 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.

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

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

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

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

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

35. 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.

36. 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.

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

38. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія*. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

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

40. Смірнов О.А., Коноплицька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

41. 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).

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

43. Дреєва Г.М., Смірнов О.А., Дреєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. *Центральноукраїнський науковий вісник. Технічні науки*. № 1(32). с. 173-183, 2019.

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

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		66

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

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

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

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

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

50. Смірнов О.А., Смірнов С.А., Дідик А.К., Дреєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. - 2016. - С. 121-127.

51. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. - Випуск 2 (46) - Х.: ХУПС - 2016. - С. 146-149.

52. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

					<b>ВКРБ-123.25.0067.00.00.ПЗ</b>	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		67

Додаток А  
(обов'язковий)

**Технічне завдання**

**Зміст**

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

					<b>ВКРБ-123.25.0067.00.00.ТЗ</b>			
Вим.	Арк.	№ документа	Підпис	Дата				
Розробив	Гужявічус В.З.				Програмне забезпечення системи динамічної хмарної платформи на основі сервісів IaaS	Літ.	Аркуш	Аркушів
Перевірів	Лисенко І.А.					Б	1	6
Н. Контр.	Коваленко А.С.				ЦНТУ КІ-21-1			
Затв.	Смірнов О.А.							

## 1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи динамічної хмарної платформи на основі сервісів IaaS.

## 2 Підстава для розробки

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

## 3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи динамічної хмарної платформи на основі сервісів IaaS.

## 4 Джерела розробки

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

## 5 Технічні вимоги

### 5.1 Склад продукції

Складниками розробки є:

- вибір і обґрунтування методів реалізації проекту;

					ВКРБ-123.25.0067.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		2

- розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;
- розробка програми, що реалізує спроектовані алгоритми роботи системи.

## 5.2 Показники призначення

Система повинна забезпечувати:

- системи динамічної хмарної платформи на основі сервісів IaaS;
- 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.

					<b>ВКРБ-123.25.0067.00.00.ТЗ</b>	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		3

## 5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

## 5.7 Вимоги до складу та параметрів технічних засобів

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

## 5.8 Вимоги до інформаційної і програмної сумісності

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

### 5.8.1 Обладнання

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

### 5.8.2 Мова програмування

Середовище Visual C#.

					ВКРБ-123.25.0067.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		2

### 5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

### 5.8.4 Вихідні дані

Робоча програма.

## 6 Вимоги до програмної документації

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

## 7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 67 аркушів.

## 8 Етапи розробки

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

					<b>ВКРБ-123.25.0067.00.00.ТЗ</b>	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		5

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

## 9 Порядок контролю та приймання

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

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

					ВКРБ-123.25.0067.00.00.ТЗ	Арк.
Вим.	Арк.	№ документа	Підпис	Дата		6

Додаток Б  
(обов'язковий)

**Міністерство освіти і науки України**  
**Центральноукраїнський національний технічний університет**

**ЗАТВЕРДЖУЮ**

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

\_\_\_\_\_ Лисенко І.А.

*Програмне забезпечення системи динамічної хмарної платформи на основі  
сервісів IaaS*

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 56

Літера: РП

Кропивницький – 2025 року

**Файл ProcessForm.cs - вікно відображення процесів запису/читання  
та перевірки цілісності даних у дата-центрі**

```

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\_2025

## 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[] VolList
{
    get
    {
        if (!InProcessing)
        {
            return this.volList;
        }
        else
        {
            return null;
        }
    }
}

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

/// <summary>

```

```

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

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

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

                        break;
                    }

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

                        break;
                    }

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

                        break;
                    }

                    case 3:
                    {

```

```

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

        break;
    }

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

        break;
    }
}

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

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

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

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

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

#endregion Public Properties & Data

#region Data

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

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

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

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

```

```

    /// </summary>
    private String fileName;

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

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

    /// <summary>
    /// Тип кодера 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
        // 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 % progressMod1) == 0)
        &&
        (OnUpdateFileAnalyzeProgress != null)
    )
    {
        OnUpdateFileAnalyzeProgress(((double) (eccNum + 1) /
(double) (this.dataCount + this.eccCount)) * 100.0);
    }

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

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

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

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

        return;
    }
}

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

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

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

}

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

```

```

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

    // Обчислюємо відсоток "" альтернативних томів, що вижили, 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

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

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

        #endregion Data

        #region Construction & Destruction

        /// <summary>
        /// Конструктор декодера за замовчуванням
        /// </summary>
        public 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
    {
        //...указуємо на помилку конфігурації
        this.configIsOK = false;
    }

    return this.configIsOK;
}

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

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

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

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

            data[i] = mulSum;

```

```

        } 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
звратної
if (pivot == 0)
{
    //...указуємо на помилку конфігурації
    this.configIsOK = false;

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

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

    return false;
}

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

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

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

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

        int idx = i_n + j;

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

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

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

```

```

        int i_n = (i * this.n);

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

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

    // Якщо є передплата на делегата відновлення прогресу -...
    if (
        ((k % progressMod1) == 0)
        &&
        (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-ой рядка
    int i_n = i * this.n;

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

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

```

стовпець

```

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

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

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

        return;
    }
}

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

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

    // Залежно від типу 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 її
вихідних елементів
            // (для прискорення множення матриці на вектор)
            this.FLog[i_n + j] = this.eGF16.Log(this.D[((this.n
+ i) * this.n) + j]);
        }
    } else
    {
        // Формування рядка в матриці кодування
        for (int j = 0; j < this.n; j++)
        {
            int idx = i_n + j;

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

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

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

```



## Файл About.cs - вікно довідки про програму

```

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

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

        #region Windows Form Designer generated code

        /// <summary>
        /// Необхідний метод для підтримки розробника - не модифікується
        /// зміст цього метода використовується редактором коду.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.developersListBoxdevelopersListBox = new
System.Windows.Forms.ListBox();
            this.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[]
{
                "БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА",
                "",
                "На тему:",
                "",
                "Програмне забезпечення системи динамічної хмарної платформи на
основі сервісів IaaS",
                "",
                "",
                "Керівник: Лисенко І.А.",
                "",
                "Розробив: студент Гужявічус Вітаутас Зігмундас",
                "                гр. KI-21-1",
                "",
                "М. Кропивницький 2025"});
            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;

}
}

```