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 адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Трущ М.О.
«____» _____ 2025 р.

Керівник проекту
доктор філософії (PhD)
_____ Усік П.С.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *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

Трущу Микиті Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій*

2. Керівник роботи *Усік Павло Сергійович, доктор філософії (PhD)*
(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 адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій*

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
_____ Усік П.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Трущ М.О.
(прізвище та ініціали)

АНОТАЦІЯ

Трущ М.О. Програмне забезпечення системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій. 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 адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

Метою розробки є програмне забезпечення системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

Результат роботи – програмна реалізація системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

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, рівноважне кодування

ABSTRACT

Trushch M.O. Software for the adaptive equilibrium coding system based on binomial numbers for cloud technologies. 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, which is intended for the adaptive equilibrium coding system based on binomial numbers for cloud technologies.

The purpose of the development is the software for the adaptive equilibrium coding system based on binomial numbers for cloud technologies.

The result of the work is the software implementation of the adaptive equilibrium coding system based on binomial numbers for cloud technologies.

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

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

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

The program was developed in Visual C#.

Keywords: computer engineering, equilibrium coding

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АСУ	–	автоматизовані системи управління
ІК	–	інформаційний канал
ІСХТ	–	інформаційна система, побудована з використанням технологій «хмарних обчислень
КЗ	–	канал зв'язку
СПД	–	системи передачі даних

ВСТУП

Актуальність теми. Система адаптивного рівноважного кодування на основі біноміальних чисел застосовується для підвищення швидкості передачі даних у хмарних технологіях.

Оптимізація швидкості передачі даних у хмарних обчисленнях має вирішальне значення для ефективної обробки та зберігання даних. Кілька факторів впливають на швидкість передачі даних, і розуміння цих факторів є важливим для оптимізації.

На швидкість передачі даних у хмарних обчисленнях впливає кілька факторів, зокрема пропускна здатність мережі, ємність зберігання, стиснення та шифрування даних:

– Пропускна здатність мережі: швидкість, з якою дані передаються через мережу, впливає на швидкість передачі даних. Швидша пропускна здатність мережі забезпечує швидшу передачу даних.

– Ємність накопичувача: ємність пристрою зберігання також впливає на швидкість передачі даних. Більший обсяг пам’яті може обробляти більше даних, що забезпечує більш високу швидкість передачі.

– Стиснення даних: стиснення даних зменшує їх розмір, що забезпечує більш високу швидкість передачі. Однак алгоритми стиснення можуть потребувати інтенсивних обчислень, що впливає на загальну продуктивність.

– Шифрування даних: шифрування даних додає додатковий рівень безпеки, але може сповільнити швидкість передачі даних. Вибір алгоритму шифрування та розміру ключа впливає на швидкість передачі даних.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

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

– Огляд існуючих систем адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

– Дослідження системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

– Програмна реалізація системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Для оптимізації швидкості передачі даних у хмарних обчисленнях можна застосувати кілька стратегій:

– Використовуйте високошвидкісні мережі: використовуйте високошвидкісні мережі, такі як 10GbE або 100GbE, для швидкої передачі даних.

– Оптимізуйте ємність сховища: забезпечте достатню ємність для обробки великих наборів даних і забезпечення вищої швидкості передачі.

– Застосуйте стиснення даних: використовуйте ефективні алгоритми стиснення, щоб зменшити розмір даних і прискорити час передачі.

– Виберіть відповідне шифрування: виберіть алгоритми шифрування та розміри ключів, які збалансують безпеку та продуктивність.

– Використовуйте спеціальні функції хмари: використовуйте спеціальні функції хмари, такі як багатокомпонентне завантаження Amazon S3 або відновлювані завантаження Google Cloud Storage, щоб оптимізувати швидкість передачі даних.

Застосування найкращих практик може допомогти оптимізувати швидкість передачі даних у хмарних обчисленнях:

– Контролюйте пропускну здатність мережі: регулярно перевіряйте пропускну здатність мережі, щоб переконатися, що вона відповідає вимогам передачі даних.

– Запровадження кешування даних: використовуйте кешування даних.

1.2 Область застосування

Областю застосування є хмарні технології. Хмарна передача даних – це процес переміщення даних у хмарне обчислювальне середовище та з нього. Існують різні потреби, які задовольняють хмарні передачі: один тип хмарної передачі даних бере дані з локального локального центру обробки даних і переносить їх у хмару. Інший тип передачі переміщує дані з однієї хмарної платформи на іншу. Третя бере дані з хмарної платформи та переносить їх у локальний центр обробки даних.

Основна перевага хмарних рішень для зберігання даних полягає в розміщенні даних у найефективніший спосіб. Зазвичай організації обирають використання хмарних служб для передачі даних або зберігання та доступу до них, виходячи з вартості, продуктивності та безпеки. Загалом, хмарне сховище забезпечує більшу еластичність, самообслуговування та резервування, що робить його чудовим вибором для організацій.

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

Skyplane

Передача даних у хмарі є повільною та дорогою. Передачі за допомогою типових інструментів CLI, таких як aws s3 забезпечують швидкість передачі всього до 20 МБ/с (нижче, ніж середня швидкість широкосмугового зв'язку в США). Крім того, хмарні передачі дуже дорогі; копіювання набору даних розміром 100 ГБ може коштувати до 14 доларів через надзвичайно високу плату за вихідні дані хмари.

Під час передачі набору даних розміром 64 ГБ Skyplane працює в 160 разів швидше, ніж rsync, і в 110 разів швидше, ніж AWS DataSync.

Skyplane – це інструмент розробника з відкритим кодом для передачі даних між хмарними сховищами об'єктів. Skyplane у 164 рази швидший за rsync і в 113 разів швидше за AWS DataSync. Те, що займало б майже половину робочого дня, тепер займає менше двох хвилин. Це менше часу, ніж потрібно для читання цього блогу!

Як ми можемо зробити перекази швидшими та дешевшими?

За останнє десятиліття, як дослідники в Берклі, наші дослідження щодо оптимізації продуктивності робочих навантажень із інтенсивним використанням даних призвели до Apache Spark і проекту Ray. Ці системи були розроблені в той час, коли набори даних переважно жили в одному регіоні в одній хмарі. все частіше,вузьке місце є*передача даних*між хмарними регіонами та хмарними провайдерами.

Ми вважаємо, що передавання даних має бути швидшим, дешевшим і

універсальним у будь-якій хмарі. Дослідження нашої лабораторії в області хмарних обчислень і мереж привели нас до створення Skyplane, який:

– до 110 разів швидше, ніж найкращі хмарні служби передачі, такі як AWS DataSync;

– до 3,8 разів дешевше, ніж існуючі безкоштовні інструменти;

– універсальна підтримка в усіх трьох публічних хмарах (AWS, Azure та GCP).

Сьогодні переміщення даних між хмарами

Існує безліч інструментів, які допомагають передавати дані через Інтернет між хмарними регіонами або хмарними провайдерами, однак кожен має обмеження:

1. **rsync** пропонує простий інструмент CLI, який копіює дані з однієї віртуальної машини на іншу, одночасно застосовуючи дельта-стиснення, щоб уникнути надлишкових копій даних, уже знайдених у місці призначення. Подібним чином хмарні провайдери пропонують CLI на зразок aws s3 срабо gsutil cp.

2. **Хмарні сервіси передачі даних,** як-от AWS DataSync і GCP Data, можуть використовувати гнучкість хмари для виконання передачі з високою пропускною здатністю. Ці служби зазвичай надають пріоритет підтримці передачі даних *у* хмару провайдера, їм бракує універсальності. Вони також можуть стягувати високу плату за вихід і обслуговування (наприклад, 1,25 доларів США за 100 ГБ, переміщених для DataSync).

Наша мета з Skyplane – надати хмарний інструмент для передачі даних, який є (1) *швидким* із реальною високою швидкістю передачі, (2) *недорогим,* щоб мінімізувати вплив надзвичайної плати за вихід, і (3) *універсальним,* оскільки підтримує всі основні публічні хмари.

Давайте перемістимо набір даних розміром 64 ГБ як один великий файл між двома регіонами в AWS (ap-southeast-3 до eu-west-3). Ми можемо зробити це за допомогою Skyplane CLI за допомогою:

```
$ skyplane cp -n 8 s3://srcbucket/64gb.tar.gz s3://dstbucket/
```

Результат? У 110 разів швидше передавання, ніж AWS DataSync і rsync).

Skyplane оптимізує швидкість передачі за допомогою мережевих досліджень Каліфорнійського університету в Берклі. Skyplane створює накладену мережу поверх хмар, щоб він міг *автоматично обходити перевантажені хмарні мережі*. Він також використовує паралелізм, розділяючи передачу даних через багато конвеєрних з'єднань TCP, а також через кілька віртуальних машин. Ми представимо глибоке занурення у внутрішню роботу Skyplane у наступній публікації блогу. Дізнайтеся більше про архітектуру Skyplane у наших документах.

Комісії за вихід даних домінують у витратах на хмару з надзвичайно високими зборами за вихід з усіх трьох публічних хмар. Переміщення 220-гігабайтного дампа англійської Вікіпедії безпосередньо за допомогою rsync з однієї віртуальної машини обійдеться лише в 4,47 доларів США за вихідну плату. Окрім вихідної комісії, такі інструменти, як AWS DataSync, стягують комісію за обслуговування переказів, таким чином підвищуючи вартість до 7,27 доларів США.

З використанням Skyplane передача 220 ГБ коштуватиме лише 1,17 долара. Це в 6,2 рази дешевше, ніж AWS DataSync, із середньою вихідною ціною 0,005 дол. США/ГБ.

Skyplane є безкоштовним і має відкритий вихідний код, тому не вимагає додаткової плати за обслуговування, на відміну від таких служб, як AWS DataSync. Skyplane зменшує витрати на вихід, прозоро стискаючи ваші дані без уповільнення передачі за допомогою стиснення LZ4. Крім того, оптимізатор Skyplane автоматично вибирає найкращий мережевий маршрут, щоб мінімізувати витрати, забезпечуючи хорошу пропускну здатність.

Універсальна підтримка будь-якої хмари

Skyplane розроблено таким чином, щоб не залежати від хмари, і наразі підтримує передачу через AWS, GCP і Azure. Існуючі платформи передачі даних, які пропонують хмари, не підтримують усі джерела та призначення.

Якщо Skyplane не підтримує вибрану вами хмару, можна додати додаткових постачальників хмар, просто інтегрувавши їхні API автентифікації та сховища об’єктів. Легко підтримувати нову хмару; надішліть запит на отримання в менш ніж 500 рядках коду, щоб додати нового хмарного постачальника.

Skyplane підтримує передачу між будь-якою загальнодоступною хмарою в будь-яку іншу велику публічну хмару

Skyplane є відкритим кодом на Github і є проектом, який підтримується спільнотою. Після входу в хмари через AWS, GCP або Azure CLI налаштування Skyplane займає менше 5 хвилин.

```
# встановити skyplane і налаштувати його за допомогою хмар
$ pip install skyplane
$ skyplane init# копіювати файли з сегмента AWS S3 у сегмент GCP GCS
$ skyplane cp -r s3://… gs://…# скопіювати змінені файли з S3 до GCS
$ skyplane sync s3://… gcs://…
```

FileCatalyst

Оскільки все більше додатків переходять на SaaS або хмарні моделі розгортання, t тут потрібен високопродуктивний доступ від цих додатків. Перенесення сховища даних із локального в хмару може бути складним завданням. Інтеграція високопродуктивної передачі файлів у ці хмарні програми та програми SaaS може бути іншим. FileCatalyst вирішує обидві проблеми за допомогою готових інструментів для легкої міграції та великого набору програмного забезпечення (SDK) для вбудовування технології в існуючі програми. Зрештою, FileCatalyst – це швидший протокол, який дозволяє легко завантажувати оголошення та забезпечує повну інтеграцію хмарного сховища даних.

Додаткові переваги FileCatalyst:

– Прискорена глобальна доставка: організації можуть доставляти медіафайли високої роздільної здатності на швидкості до 10 Гбіт/с. Загалом це скорочує виробничі цикли та робочі процеси.

– Повний контроль пропускної здатності: навіть за поганих умов мережі організації можуть регулювати передачу за важливістю та уникати збоїв у

мережі.

– Економія часу та грошей: за допомогою рішень FileCatalyst можна уникнути всіх зависань, пов’язаних із передачею фізичних медіа.

– Передача зростаючих файлів: файли можна передавати під час їх кодування, щоб покращити зручність використання та якість швидкості передачі.

– Незрівнянна безпека: організації можуть забезпечити конфіденційність передачі файлів за допомогою галузевих стандартних сертифікатів шифрування, таких як AES і SSL.

– Попередній перегляд із низькою роздільною здатністю: у рішеннях FileCatalyst організації можуть переглядати попередні перегляди з низькою роздільною здатністю, щоб переконатися, що вони отримали доступ до потрібного вмісту.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Синтаксис Visual C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C++ або Java з легкістю взнають синтаксис із фігурними дужками, характерний для мови Visual C#. Розроблювачі, що знають кожну із цих

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

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

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

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

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

– Атрибути з декларативними метаданими про типи під час виконання.

– Вбудовані коментарі XML-документації.

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

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

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

Архітектура платформи .NET Framework

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

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

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

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

Взаємодія між мовами є ключовою особливістю .NET Framework. Оскільки код IL, створюваний компілятором Visual C# відповідає специфікації CTS, код IL на основі Visual C# може взаємодіяти з кодом, створюваним версіями мов Visual Basic, Visual C++, Visual J# платформи .NET Framework і ще більш ніж 20 CTS-сумісних мов. В одному складанні може бути кілька модулів, написаних на різних мовах платформи .NET Framework, і типи можуть посилатися один на одного, як якби вони були написані на одній мові.

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

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) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Хмарна інфраструктура, завдяки стрімкому розвитку 5G, периферійних обчислень і хмарних технологій, відкриває значні можливості для постачальників послуг зв’язку (CSP) для підвищення ефективності, підтримки інноваційних послуг і розкриття нового бізнес-потенціалу. Дізнайтеся, як найкраще використовувати, створювати та трансформувати хмарну інфраструктуру, щоб відповідати вимогам мереж і послуг наступного покоління за підтримки перевірених рішень телекомунікаційного рівня.

Хмарна інфраструктура – це поєднання апаратного, програмного забезпечення, мереж, технологій зберігання та віртуалізації, які утворюють основу сучасних телекомунікаційних мереж. Він надає необхідні ресурси, необхідні для створення гнучких хмарних середовищ і керування ними, дозволяючи CSP надавати послуги більш ефективно та в масштабі.

Архітектура хмарної інфраструктури дозволяє динамічно розподіляти ці ресурси між програмами. Це критично важливо для керування мережевими функціями, системами підтримки операцій (OSS), системами підтримки бізнесу (BSS) та іншими IT-службами. Гнучкість хмарної інфраструктури дозволяє CSP зменшити операційні витрати та підвищити продуктивність, особливо під час підготовки до розгортання 5G.

Хмарна інфраструктура зазвичай надається через моделі «як послуга», такі як інфраструктура як послуга (IaaS), платформа як послуга (PaaS) і програмне забезпечення як послуга (SaaS). Це надає CSP гнучкість для прийняття індивідуальних рішень на основі їхніх конкретних потреб.

Традиційно CSP покладалися на виділену локальну інфраструктуру для кожної програми, використовуючи функції фізичної мережі та монолітну

архітектуру з тісно пов’язаними компонентами. Цей підхід призвів до складних ручних операцій і обмеженої масштабованості, що ускладнювало адаптацію до мінливих вимог ринку.

Оскільки галузь вимагала більшої гнучкості та ефективності витрат, запровадження віртуалізації мережі дозволило CSP перейти до більш динамічної та масштабованої інфраструктури. Завдяки відокремленню апаратних засобів, таких як обчислювальні, мережеві та ресурси зберігання, від додатків, які їх використовують, віртуалізація дозволила CSP створити спільний пул ресурсів, які можна було розподіляти за потреби, що призвело до більш ефективного використання інфраструктури.

Подорож хмарної інфраструктури тепер продовжується впровадженням власних хмарних технологій, що представляє наступну еволюцію в архітектурі. Дозволяючи CSP розгортати додатки за допомогою контейнерів і мікросервісів безпосередньо в інфраструктурі «голого металу», хмарні власні технології ще більше підвищують гнучкість, роблячи розгортання нових сервісів швидшим і ефективнішим. Це також стимулює інновації, дозволяючи операторам зв’язку швидко реагувати на зміни ринку та масштабуватись за потреби, особливо під час підготовки до розгортання 5G і далі.

Ключові переваги хмарної інфраструктури телекомунікацій

Хмарна інфраструктура забезпечує спільний доступ до ресурсів і пропонує уніфіковану структуру, яка спрощує керування послугами та знижує витрати. У результаті CSP можуть динамічно розподіляти ресурси, ефективно масштабувати та оптимізувати операції – необхідні можливості для збереження конкурентоспроможності в телекомунікаційному середовищі, що швидко розвивається:

– Зменшити витрати. Спільне використання ресурсів між додатками для оптимізації робочих навантажень і зниження витрат на інфраструктуру шляхом мінімізації залежності від спеціального обладнання.

– Оптимізовані операції. Керуйте єдиною інфраструктурою за допомогою

централізованих інструментів і автоматизуйте робочі процеси, щоб спростити операції, прискорити розгортання та зменшити складність.

– Швидший вихід на ринок. Розгортайте програми за допомогою попередньо налаштованих хмарних середовищ і автоматизованих робочих процесів.

– Прискорені інновації. Використовуйте хмарну інфраструктуру та API, щоб дозволити розробникам швидко створювати, тестувати та розгортати нові програми для служб нового покоління.

– Динамічне масштабування. Збільшуйте або зменшуйте масштаб ресурсів і послуг, забезпечуючи гнучкість керування коливаннями попиту та підтримку розгортання 5G або інших послуг із високим попитом.

– Підвищення гнучкості. Використовуйте власні хмарні технології, щоб швидко адаптуватися до мінливих вимог, забезпечуючи динамічне розгортання послуг.

Важливо відзначити, що ступінь цих переваг залежить від різних факторів, включаючи хмарну архітектуру, конкретні технології та ефективність управління інфраструктурою.

Як орієнтуватися в ландшафті, що розвивається: ключові міркування

Чи готові ви зробити наступний крок у трансформації вашої інфраструктури, щоб відповідати вимогам мереж і послуг нового покоління? Перш ніж почати цю подорож, є кілька важливих областей, які CSP повинні розглянути для успішної трансформації.

– Варіанти використання та бізнес-стратегія для зростання.

– Характеристики застосування.

– Можливість наскрізного керування життєвим циклом (LCM) і підтримки.

– Внутрішня компетенція та модель роботи.

– Гнучкість проти мінімізації складності.

– Багатохмарна стратегія та відкритість.

– Телекомунікаційні правила та суверенітет даних для окремих країн.

Оскільки хмарна інфраструктура продовжує розвиватися, постачальники послуг також повинні узгоджувати свої хмарні стратегії з технологічними змінами, які рухають галузь. Еволюція від віртуалізованих середовищ до власної хмарної архітектури відкриває нові рівні ефективності та гнучкості, уможливлюючи інноваційні випадки використання, які раніше були недоступні.

У той же час вибір моделей розгортання розвивається. Постачальники послуг тепер можуть застосовувати різні моделі від приватних до публічних і гібридних хмарних підходів Hyperscale Cloud Provider (HCP), забезпечуючи більшу гнучкість, масштабованість і операційні переваги. Ці моделі дозволяють адаптувати інфраструктуру відповідно до операційних цілей і стратегічних потреб.

Давайте дослідимо, як ці досягнення впливають на вибір хмарної інфраструктури та як CSP можуть використовувати їх, щоб залишатися попереду в цьому динамічному ландшафті.

Варіанти розгортання: виберіть між приватною, публічною хмарою HCP або гібридною хмарою

Еволюція моделей хмарного розгортання була сформована потребою в більшій гнучкості, продуктивності та ефективності в управлінні інфраструктурою. Постачальники послуг обирають ці моделі на основі своїх стратегічних цілей і вимог до операційної моделі, забезпечуючи узгодженість із загальними бізнес-цілями. У телекомунікаційній галузі ці вимоги є ще більш критичними, де низька затримка, високий рівень безпеки та стійкість даних є важливими для підтримки операцій телекомунікаційного рівня.

Для багатьох CSP приватна хмара залишається переважним вибором для досягнення цієї мети, пропонуючи необхідний контроль, безпеку та налаштування. Багатохмарні стратегії, де кілька приватних хмар оптимізовані для різних робочих навантажень, також стають все більш поширеними. У той же час CSP застосовують гібридні хмарні стратегії, які об’єднують використання

приватної та загальнодоступної хмари HCP для некритичних робочих навантажень, пропонуючи додаткову гнучкість і масштабованість, де це необхідно:

– Приватна хмара.

– Публічна хмара HCP.

– Гібридна хмара.

Використання хмарних технологій для гнучкої, готової до майбутнього інфраструктури

Внутрішня хмарна інфраструктура забезпечує гнучкість підтримки як хмарних функцій (CNF), так і функцій віртуальної мережі (VNF), особливо в розгортанні «голого металу», коли CSP модернізують свої мережі. Очікується, що VNF залишаться частиною мережевих екосистем протягом деякого часу, можливість співіснувати з CNF на єдиній платформі дає змогу CSP безперебійно поєднувати застарілі функції та функції наступного покоління. Такий підхід дозволяє здійснити поступовий перехід до власних хмарних технологій із низьким рівнем ризику, адаптацію до мінливих вимог до мережі, зберігаючи при цьому стабільність роботи.

Розгортання як CNF, так і VNF безпосередньо на інфраструктурі «голого металу» усуває потребу в додаткових рівнях віртуалізації, що може оптимізувати використання апаратного забезпечення та значно знизити загальну вартість володіння. Завдяки оптимізації архітектури та мінімізації зайвих рівнів, CSP можуть досягти значної економії коштів і підвищення продуктивності всієї своєї мережевої інфраструктури.

Ця ефективність закладає основу для ефективного управління через оркестровку контейнерів. Такі платформи, як Kubernetes, відіграють вирішальну роль, керуючи контейнерними функціями в центральних, периферійних і приватних хмарах. Kubernetes автоматизує масштабування, керування ресурсами та відновлення після збоїв, забезпечуючи високу доступність і оптимальну продуктивність, особливо в 5G і периферійних обчислювальних середовищах.

Зі зростанням поширення моделі розподіленої хмари роль оркестровки контейнерів стає ще важливішою. Досягнення операційної ефективності та належного рівня функціональності в цьому складному середовищі вимагає централізованої та єдиної системи автоматизації. Така система має важливе значення для CSP для ефективної роботи своїх мереж, дозволяючи їм керувати ресурсами в кількох хмарних розташуваннях. Це забезпечує постійну продуктивність, відповідність і безпеку, одночасно спрощуючи операційні складності.

Подолання ключових викликів

Оскільки CSP продовжують орієнтуватися в складнощах впровадження базових і хмарних технологій 5G, вони стикаються з кількома ключовими проблемами. Подолати ці перешкоди може бути важко без правильного керівництва та підтримки.

Забезпечення чіткої звітності під час операцій для LCM та підтримки

Завдання CSP: Управління життєвим циклом складних функцій базової мережі 5G стає складним, особливо коли ми маємо справу з різноманітною екосистемою постачальників і технологій. Проблема полягає в оптимізації операцій і забезпеченні узгодженої підтримки в усіх фрагментованих системах.

Потенційний вплив: без чіткої підзвітності та автоматизованого LCM за допомогою методологій, таких як DevOps, і таких практик, як CI/CD, CSP можуть зіткнутися з операційними ризиками, збільшенням часу простою, фрагментованою підтримкою та непослідовними оновленнями, що призведе до вищих операційних витрат.

Найкращі практики: застосування практик DevOps і впровадження автоматизованих конвеєрів CI/CD оптимізують LCM, забезпечуючи послідовне розгортання та послідовне розгортання та оновлення без перерв у роботі. Оновлення програмного забезпечення під час роботи (ISSU) дозволяє оновлювати в реальному часі без простоїв, забезпечуючи високу доступність і знижуючи операційні ризики.

Використання одного постачальника як для додатків, так і для інфраструктури оптимізує роботу та гарантує повну інтеграцію всіх компонентів. Цей підхід спрощує LCM, дотримуючись узгоджених графіків оновлення, забезпечуючи сумісність компонентів і надаючи комплексну підтримку для всієї системи.

Спрощення інтеграції фрагментованих компонентів

Завдання CSP: плавна інтеграція різних компонентів із різними рівнями сумісності може бути надзвичайно важкою, особливо в складних 5G і хмарних середовищах.

Потенційний вплив: керування різноманітними компонентами часто призводить до затримок, збільшення витрат на розгортання та уповільнення часу виходу на ринок, що безпосередньо впливає на прибутковість.

Найкращі практики: багато CSP вважають, що перевірені системою попередньо інтегровані рішення допомагають зменшити складність, забезпечуючи безперебійну інтеграцію компонентів. Хоча деякі все ще вважають за краще проводити власні тести на сумісність, щоб переконатися, що всі компоненти відповідають їхнім унікальним екосистемам, цей підхід може створити додаткові вимоги до складності та ресурсів. У таких випадках використання схем, спільно перевірених програмою та постачальниками хмарних технологій, може забезпечити плавну інтеграцію та зменшити складність.

Оптимізація використання ресурсів

Завдання CSP: ефективне керування обчислювальними та мережевими ресурсами відповідно до попиту, водночас збалансовуючи продуктивність і економічну ефективність може бути складним завданням.

Потенційний вплив: неефективне управління ресурсами може призвести до недостатнього використання інфраструктури, марного використання потужностей і зростання операційних витрат, що зрештою вплине на якість послуг.

Найкращі практики: використання хмарної інфраструктури з

уніфікованою архітектурою в централізованих, периферійних і приватних мережах може допомогти CSP оптимізувати використання ресурсів. Багато постачальників реалізують Kubernetes на «голому металі», щоб зменшити вимоги до серверів і досягти економічно ефективного, масштабованого управління ресурсами. Крім того, послідовне управління обладнанням забезпечує плавний перехід між різними поколіннями технологій, підтримуючи довгострокову ефективність.

Однією з проблем, з якою стикаються організації під час хмарної міграції, є концепція тяжіння даних. Гравітація, у цьому контексті, – це ідея про те, що зі збільшенням розміру даних до них буде притягнуто більше програм і послуг. Оскільки організації потрібно більше даних, їй знадобиться більше структури для їх розміщення, а потім більше даних і так далі. Зрештою, чим більше даних, тим більше людина «застрягає» з постачальником.

Ось кілька переваг і недоліків хмарного сховища, які слід враховувати :

Переваги хмарного сховища:

– Хмара є одним із найбільш економічно ефективних методів, які може використовувати організація.

– Зі збільшенням пропускної здатності мережі зростає і швидкість, з якою дані передаються в хмару. Цей елемент особливо важливий для віддаленої робочої сили, що дає будь-якій організації можливість ефективно та ефективно передавати дані з будь-якої точки світу, не створюючи вузьких місць передачі даних.

– Сховище C Loud забезпечує підвищення гнучкості, зниження витрат і надійнішу безпеку.

Недоліки хмарного сховища:

– Проблема хмарного сховища полягає в тому, щоб знайти обхідний засіб, який допоможе підключити ваш традиційний сервер і перемістити дані в хмарне середовище ; наприклад, ваш відер S3 b.

– Швидкий доступ до даних неможливий без швидкого підключення до

Інтернету та / або додаткової реалізації швидкої передачі файлів.

Ключ до вирішення основної проблеми хмарного сховища даних полягає в максимізації швидкості завантаження та завантаження без створення вузьких місць передачі даних, які впливають на використання вашого хмарного сховища.

3 основні міркування щодо передачі даних у хмару

Все більше організацій починають зберігати свої дані в хмарі, а не лише на локальних серверах. Але як вони переміщують дані у свої хмарні програми, використовують їх і переносять із хмари? Працюючи з хмарними програмами, важливо враховувати практичні аспекти передачі даних у хмарі, зокрема:

1. Як ви будете переміщувати ці дані в хмару та з неї.

2. Як ви отримаєте доступ до нього після перенесення в хмару.

3. Що робити, якщо ви працюєте з такими обмеженнями, як великі набори даних, вимоги щодо короткого часу доступу або віддалені джерела та призначення.

Розроблене програмне забезпечення – це швидке рішення для передачі файлів, яке інтегрується з кількома найпопулярнішими рішеннями для зберігання об’єктів, щоб забезпечити вам гнучкість у підході. Використання програмного забезпечення для прискорення файлів для передачі хмарних даних допоможе організаціям вивчити труднощі, з якими вони зіткнуться, і способи їх подолання.

Хмарні програмні рішення масово впроваджуються, головним чином у галузях, які покладаються на швидкий доступ до даних, таких як медіа, телемовлення та розваги. У той час як у галузях із суворими вимогами дотримання вимог і безпеки, як-от банківська справа та охорона здоров’я, ці рішення були прийняті нещодавно. Розроблене програмне забезпечення допомагає організаціям будь-якого типу з їхніми робочими процесами, дозволяючи користувачам швидко переміщувати, редагувати та надсилати дані зацікавленим сторонам.

Прискорення руху відеоресурсів для ЗМІ та розваг

У випадку медіа-індустрії передача відеовмісту через хмару розширена можливостями розробленого програмного забезпечення. Монтажні студії можуть переміщувати відеоресурси в хмару для редагування, нотування та зберігання, зокрема в основних програмах для редагування відео, таких як Avid і Object Matrix. Крім того, високопродуктивне обчислення Розроблене програмне забезпечення може зберігати зображення високої роздільної здатності, які потрібно проаналізувати або відредагувати в хмарі.

Чи безпечна передача та зберігання даних у хмарі?

Безпека ваших даних є надзвичайно важливою, а безпека передачі та зберігання даних у хмарі настільки ж надійна, наскільки безпечні інструменти, які ви використовуєте

3.2 Розробка структурної схеми

У сучасну цифрову епоху ефективне хмарне зберігання та передача даних є надзвичайно важливими для малого бізнесу. Ось кілька практичних порад, які допоможуть вашим клієнтам підвищити швидкість хмарного сховища та передачі даних.

Виберіть правильного постачальника хмарних сховищ

Вибір надійного постачальника хмарних сховищ є першим кроком до оптимізації. Шукайте постачальників, які пропонують можливості високошвидкісної передачі даних, надійні функції безпеки та можливості масштабованого зберігання. Серед популярних варіантів – Amazon Web Services (AWS), Google Cloud і Microsoft Azure. Оцініть їх ефективність, гарантії безвідмовної роботи та підтримку клієнтів, щоб прийняти зважене рішення.

Оптимізуйте розміри та формати файлів

Великі файли можуть значно сповільнити швидкість передачі даних. Заохочуйте своїх клієнтів стискати файли перед завантаженням їх у хмару. Такі

інструменти, як WinRAR, 7-Zip і вбудовані функції стиснення в операційних системах, можуть допомогти зменшити розміри файлів без шкоди для якості. Крім того, використання ефективних форматів файлів, таких як JPEG для зображень і MP4 для відео, може ще більше підвищити швидкість передачі.

Запровадити дедуплікацію даних

Дедуплікація даних – це техніка, яка усуває повторювані копії даних, зменшуючи вимоги до зберігання та пришвидшуючи передачу даних. Зберігаючи лише унікальні екземпляри даних, компанії можуть заощадити пропускну здатність і підвищити загальну ефективність. Багато постачальників хмарних сховищ пропонують вбудовані функції дедуплікації, або ви можете скористатися інструментами сторонніх розробників, як-от Veeam або Veritas.

Використовуйте мережі доставки вмісту (CDN)

CDN розподіляють вміст між кількома серверами по всьому світу, забезпечуючи швидший доступ до даних для користувачів незалежно від їхнього місцезнаходження. Кешуючи дані ближче до кінцевих користувачів, CDN можуть значно зменшити затримку та підвищити швидкість завантаження. Такі служби, як Cloudflare, Akamai та Amazon CloudFront, є популярними варіантами CDN, які можна інтегрувати з хмарними рішеннями для зберігання.

Відстежуйте та керуйте використанням пропускної здатності

Обмеження пропускної здатності можуть перешкоджати швидкості передачі даних. Заохочуйте своїх клієнтів стежити за використанням пропускної здатності та визначати час пікового використання. Впровадження налаштувань якості обслуговування (QoS) на маршрутизаторах може визначити пріоритетність критичних передач даних і ефективніше розподілити пропускну здатність. Крім того, планування великих передач даних у непіковий час може допомогти уникнути перевантаження мережі.

Використовуйте багатопотокові перекази

Багатопотокові передачі розділяють дані на менші частини та передають їх одночасно, прискорюючи загальний процес. Багато постачальників хмарних

сховищ підтримують багатопотокове передавання, і такі інструменти, як FileZilla, Cyberduck і Rclone, можуть полегшити цей процес. Переконайтеся, що підключення до Інтернету та апаратне забезпечення ваших клієнтів можуть обробляти багатопотокові передачі для оптимальної продуктивності.

Регулярно оновлюйте програмне забезпечення та мікропрограму

Застаріле програмне забезпечення та мікропрограми можуть призвести до проблем із сумісністю та зниження швидкості передачі даних. Заохочуйте своїх клієнтів оновлювати свої операційні системи, програми для хмарних сховищ і мережеве обладнання. Регулярні оновлення часто включають підвищення продуктивності та виправлення безпеки, які можуть підвищити загальну ефективність.

Застосуйте суворі заходи безпеки

Оптимізуючи швидкість, важливо не йти на компроміс із безпекою. Переконайтеся, що ваші клієнти використовують надійні паролі, увімкніть двофакторну автентифікацію та зашифруйте конфіденційні дані перед передачею їх у хмару. Захищені протоколи передачі даних, такі як SFTP, FTPS і HTTPS, можуть захистити дані під час передачі та запобігти несанкціонованому доступу.

Навчайте співробітників передовим практикам

Людська помилка часто може бути вузьким місцем у процесах передачі даних. Проводьте регулярні тренінги, щоб навчити співробітників найкращим практикам хмарного зберігання та передачі даних. Підкресліть важливість дотримання політики компанії, використання безпечних з'єднань і уникнення непотрібної передачі даних.

Регулярно перевіряйте та оптимізуйте сховище

Проводьте регулярні перевірки хмарного сховища ваших клієнтів, щоб виявити невикористані або зайві дані. Видалення непотрібних файлів і ефективне впорядкування даних можуть звільнити місце для зберігання та підвищити швидкість передачі. Впровадження автоматизованих інструментів для керування та очищення даних може оптимізувати процес і забезпечити оптимізацію.

Рисунок 3.1 – Структурна схема системи

Оптимізація хмарного сховища та швидкості передачі даних має важливе значення для малого бізнесу, щоб підтримувати ефективну роботу та залишатися конкурентоспроможними. Дотримуючись цих порад, ваші клієнти зможуть покращити свою хмарну інфраструктуру, зменшити затримку та підвищити загальну продуктивність. Як постачальника послуг, яким керує IT, ваш досвід і вказівки можуть значно змінити їх у досягненні цих цілей.

Система призначена для реалізації адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

До вихідних положень і понять відносять такі. У процесі передачі повідомлення по тракту прямого зв'язку інформаційного каналу (ІК) вихідне повідомлення у вигляді двійкової кодової комбінації $X = (x_1,...,x_m)$ кодується завадостійким кодом з результатом у вигляді вхідної послідовності каналу $X_к = (x_{к1},...,x_{кn})$. На вихід каналу зв'язку (КЗ) надходить кодова комбінація $Y_к = (y_{к1},...,y_{кn})$.

Якщо перехід $X_к$->$Y_к$ (одноразова передача) відбувся правильно, то $Y_к = X_к$, а у випадку неправильного переходу (через вплив завад) $Y_к$->$X_к$.

Після виявлення помилок і перепитувань вихідна послідовність $Y_к$ каналу декодується з результатом у вигляді кодової комбінації $Y = (y_1,...,y_m)$, що видається споживачу. Рівність Y = X вказує на правильну передачу.

Для оцінки результатів передачі повідомлень X споживачу використовуються ймовірності $P_п$, $P_{ош}$, $P_{оо}$ і $P_{но}$ відповідно до правильної, неправильної, виявленої та невиявленої помилкових передач [5].

Для оцінки результатів одноразової передачі по КЗ рівноважних кодових комбінацій $X_к$ (без виправлення помилок) використовуються ймовірності П і W, Z і V відповідно до правильної та неправильної передач, виявлення і невиявлення помилок [4].

3.3 Розробка функціональної схеми

У даному розділі розглянуто розроблення методів рівноважного кодування та декодування дискретних повідомлень на основі біноміальних чисел при адаптивно змінюваних значеннях параметрів рівноважного коду.

Двійкові біноміальні числа використовуються як проміжні кодові комбінації $X_б$ і $Y_б$ у двохетапних процедурах прямого перетворення X®$X_б$®$X_к$ вихідного повідомлення X у рівноважну кодову комбінацію $X_к$ і зворотного

перетворення $Y_К$®$Y_б$®Y рівноважної кодової комбінації $Y_К$ у передану споживачеві двійкову послідовність Y.

У біноміальній системі числення з параметрами n і k будь-яке біноміальне число подається двійковою кодовою комбінацією $A = a_{r-1},...,a_j,...,a_0$ довжиною r і задовольняє умови $q_0 = k$, $r < n$ або $q_0 < k$, $r-q_0 = n-k$, де q_0 – кількість одиничних розрядів. Кількісний еквівалент біноміального числа A визначається за функцією

$$F_б(A) = \sum_{j=0}^{r-1} a_j g_j , \qquad (3.1)$$

де

$$g_j = C_{n-r+j}^{k-q_{j+1}}; \quad q_{j+1} = \sum_{i=j+1}^{r} a_i ; \quad j = \overline{0, r-1}; \quad q_r = a_r = 0 .$$

На основі формули (3.14) будуються алгоритми перетворення двійкового числа у нерівномірне біноміальне та взаємно однозначного йому зворотного перетворення. Перетворення A®$X_К$ здійснюється доповненням послідовності A молодшими одиничними або нульовими розрядами, а перетворення $Y_К$®A – виключенням надлишкових молодших розрядів. Розроблено алгоритми [2] рівноважного кодування і декодування на основі нерівномірних біноміальних чисел. Доведено їхню перевагу відносно алгоритмічної складності порівняно з алгоритмами, побудованими на основі рівномірних біноміальних чисел.

Розроблено структури апаратних пристроїв рівноважного кодування і декодування на основі нерівномірних біноміальних чисел за розробленими алгоритмами, що відрізняються від відомих пристроїв того самого функціонального призначення з використанням рівномірних біноміальних чисел меншою схемотехнічною складністю і роботоздатністю при адаптивно змінюваних параметрах n і k рівноважного коду.

З огляду на переваги рівноважних кодів, з метою досягнення потрібної вірогідності передачі команд телеуправління проектом передбачається використання синтезованого за допомогою розробленого алгоритму набору з восьми рівноважних кодових комбінацій. Аналіз завадостійкості отриманих

кодових комбінацій показав, що їх використання забезпечить потрібну вірогідність передачі даних. Реалізація розробленого алгоритму формування рівноважних комбінацій, які дозволяють організувати більш ефективну передачу даних для асиметричних КЗ, здійснюється на базі програмованих контролерів типу I-7188.

Функціональна схема розробленої системи зображена на рисунку 3.2.

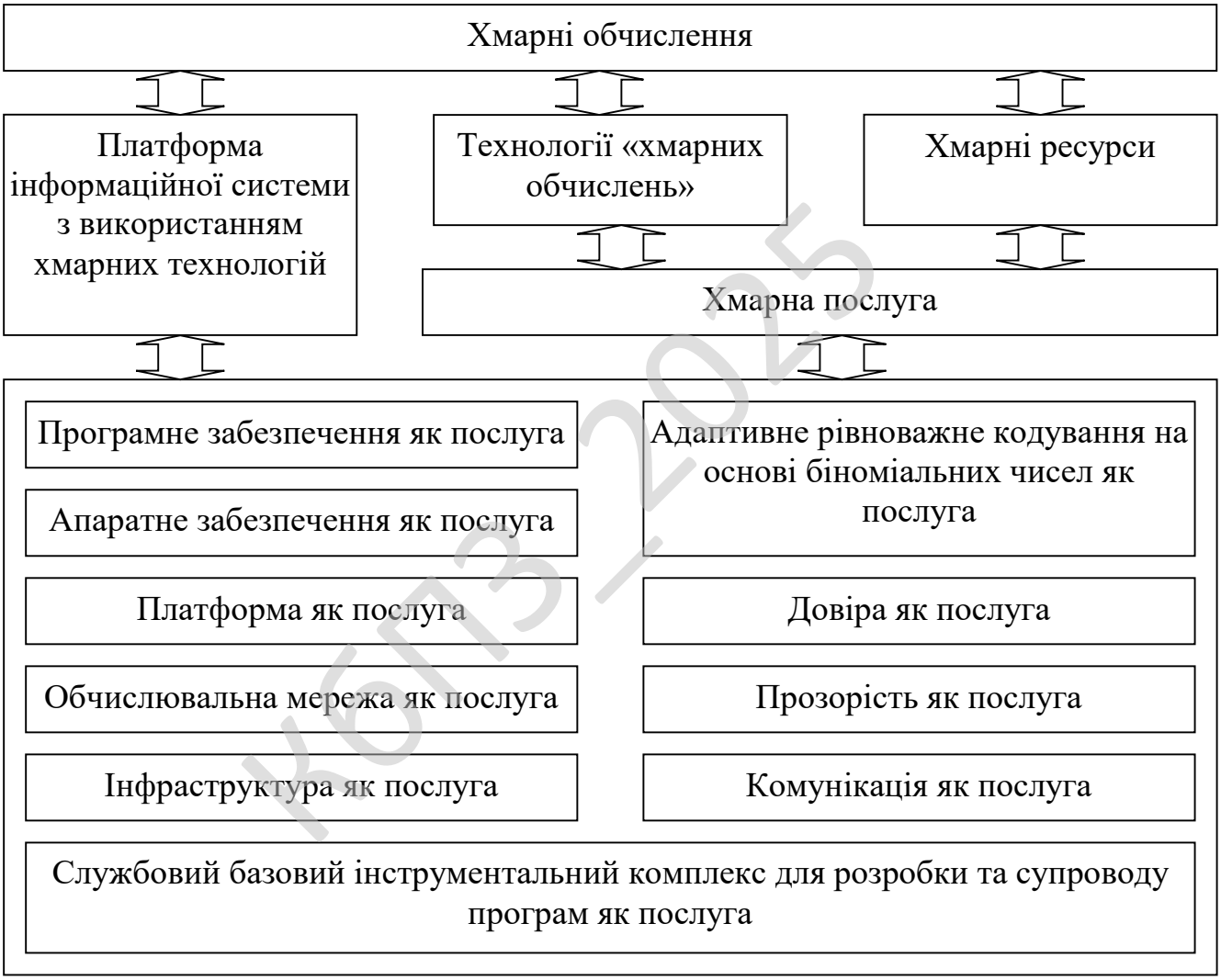

Рисунок 3.2 – Функціональна схема системи

Термінологію, що використовується на функціональній схемі, застосовну в області кодування інформації, оброблюваної за допомогою технологій «хмарних обчислень», становлять наступні терміни й відповідні їм визначення.

Хмарні обчислення – концепція надання за запитом (заявкою) мережного доступу до комп'ютерних ресурсів, що розподіляються (обчислювальним мережам, серверам, сховищам даних, програмному забезпеченню, мережним сервісам і ін.), виділюваним поза залежністю від часу доби, каналу доступу в обчислювальну мережу.

Технології «хмарних обчислень» – інформаційні технології, що реалізують концепцію «хмарних обчислень».

Хмарні ресурси – будь-які види комп'ютерних ресурсів, що розподіляються між споживачами за допомогою технологій «хмарних обчислень».

Хмарними ресурсами можуть бути, наприклад, процесорний час (обчислювальна потужність), місце в сховищах даних, обчислювальні мережі, бази даних і програмне забезпечення.

Хмарна послуга – послуга надання хмарних ресурсів за допомогою технологій «хмарних обчислень».

Хмарні послуги повинні задовольняти наступним істотним вимогам:

– самообслуговування за запитом споживачів. Споживач в однобічному порядку може змінювати обсяг надаваних йому послуг в автоматичному режимі без втручання співробітників провайдеру;

– широкополосний доступ в обчислювальну мережу. Доступ до хмарних ресурсів представляються споживачам через обчислювальну мережу за допомогою стандартних механізмів «тонкого» або «товстого» клієнтів;

– об'єднання хмарних ресурсів у єдиний загальний пул. Хмарні ресурси провайдеру поєднуються в єдиний загальний пул для обслуговування безлічі споживачів у багатозадачному режимі – різні фізичні й віртуальні хмарні ресурси динамічно виділяються й перерозподіляються відповідно до заявок споживачів;

– оперативна реакція. Обсяг надаваних споживачеві хмарних ресурсів може швидко й гнучко змінюватися (у деяких випадках – автоматично) – збільшуватися або зменшуватися. Для кінцевого споживача хмарні ресурси

провайдеру представляються нескінченними й можуть бути придбані в будь-якій кількості в будь-який час;

– вимірність. Хмарна система автоматично контролює й оптимізує ресурси, вимірюючи обсяг хмарних ресурсів на деяких рівнях абстракції відповідно до типу надаваних послуг (наприклад, сховище даних, обчислення, пропускна здатність каналу зв'язку й облікових записів користувачів).

Крім того, використання хмарних ресурсів може контролюватися й ураховуватися прозоро для провайдеру й споживача.

Інформаційна система, побудована з використанням технологій «хмарних обчислень» (ІСХТ) – інформаційна система, призначена для реалізації хмарних послуг.

Хмарний клієнт (орендар хмари) – засіб обчислювальної техніки, що входить до складу ІСХТ, за допомогою якого здійснюється одержання однієї або декількох хмарних послуг.

Хмарний сервер – розподілена обчислювальна мережа, що надає хмарним клієнтам одну або кілька хмарних послуг.

Інфраструктура хмарного сервера – інфраструктура, що включає обчислювальну мережу, сервери, операційні системи, сховища, бази даних, прикладні програми й конкретні функції програм, за винятком, можливо, обмежених обумовлених користувачем параметрів конфігурації програм.

Оператор ІСХТ (постачальник хмарних послуг) – особа, відповідальна за функціонування хмарного сервера.

Споживач хмарних послуг – особа, що здійснює доступ за допомогою хмарного клієнта до однієї або декількох хмарним послугам, надаваним хмарними серверами.

Модель хмарного розміщення – модель реалізації ІСХТ, відповідно до якої визначається приналежність операторів ІСХТ (постачальника хмарних послуг) і споживачів хмарних послуг.

У цей час використовуються наступні моделі хмарного розміщення:

– приватна хмара;

– публічна хмара;

– суспільна хмара;

– гібридна хмара.

Приватна хмара – модель хмарного розміщення, у якій оператор ІСХТ (постачальник хмарних послуг) і всі споживачі хмарних послуг належать одній організації.

Публічна хмара – модель хмарного розміщення, у якій оператор ІСХТ (постачальник хмарних послуг) і споживачі хмарних послуг належать різним організаціям.

Суспільна хмара – модель хмарного розміщення, відповідно до якого хмарні ресурси використовуються конкретним співтовариством споживачів з організацій, що мають загальні завдання.

Гібридна хмара – модель хмарного розміщення, у якій об'єднані дві й більше ІСХТ, що належать різним організаціям або типам моделей (приватним, суспільним або публічним).

Платформа ІСХТ – система програмних і програмно-апаратних засобів, що реалізують концепцію «хмарних обчислень» відповідно до моделі хмарного розміщення й видом надаваних хмарних послуг.

Основними видами хмарних послуг є:

– програмне забезпечення як послуга;

– платформа як послуга;

– інфраструктура як послуга.

Мультиарендуємість – характеристика ІСХТ, що полягає в розподілі хмарних ресурсів між безліччю хмарних клієнтів, причому частина хмарних ресурсів, надаваних хмарному клієнтові, захищена від неправомірного (несанкціонованого) доступу з боку інших хмарних клієнтів.

Способи досягнення такої ізоляції розрізняються для різних типів надаваних хмарних ресурсів, впливаючи на конфіденційність, цілісність і доступність.

Міжхмарні обчислення – концепція перерозподілу хмарних ресурсів між взаємодіючими ІСХТ на вимогу.

Службовий базовий інструментальний комплекс для розробки й супроводу програм – середовище створення, розгортання, виконання, керівництва й керування програмним забезпеченням, що реалізує одну або більше хмарних послуг.

Програмне забезпечення як послуга – хмарна послуга з надання можливості використання прикладного програмного забезпечення, розміщеного на хмарному сервері, а також зберігання результатів роботи такого програмного забезпечення.

Доступ до прикладного програмного забезпечення може бути здійснений клієнтом з використанням технології «тонкий клієнт» (таких як браузер). Споживачеві не надається можливість контролю або керування яке забезпечує роботу програм хмарної інфраструктури.

Апаратне забезпечення як послуга – хмарна послуга з надання можливості використання апаратного забезпечення хмарного сервера, для установки власного програмного забезпечення.

Платформа як послуга – хмарна послуга з надання можливості запуску в інфраструктурі хмарного сервера власних програм, створених з використанням мов і засобів програмування, підтримуваних хмарним сервером. Споживачеві не надається можливість контролю або керування інфраструктурі хмарного сервера.

Обчислювальна мережа як послуга – хмарна послуга з надання можливості використання служби мережних з'єднань і/або міжхмарних мережних з'єднань.

Інфраструктура як послуга – хмарна послуга з надання можливості використання частини інфраструктури хмарного сервера споживачем послуги для власних потреб.

Споживач може управляти роботою операційної системи, віртуальних систем зберігання даних і встановлених додатків, а також має обмежені можливості по контролі набору доступних сервісів (наприклад, міжмережний екран, DNS).

Контроль і керування основною фізичною й віртуальною інфраструктурою хмари, у тому числі мережі, серверів, вибір і завдання типів використовуваних операційних систем, систем зберігання здійснюється оператором ICXT.

Комунікації як послуга – хмарна послуга з надання можливості використання сервісів зв'язку (комунікацій) і спільної роботи в реальному масштабі часу.

Сервіси зв'язку (комунікацій) і спільної роботи включають голосове спілкування по IP-мережах, обмін миттєвими повідомленнями й відеоконференції.

Безпека як послуга – хмарна послуга з надання можливості керування безпекою по моделі аутсорсингу.

Звичайно, безпека як послуга (SaaS від англ.) включає такі додатки як антивірусне програмне забезпечення, надаване через Інтернет, проте, у рамках даної послуги може надаватися послуга керування внутрішньою безпекою зовнішньою організацією.

Довіра як послуга – хмарна послуга з надання можливості обробки службових відомостей про забезпечення безпеки користувальницької інформації в хмарі, а також захисту оператора послуг від шкідливої активності споживачів хмарних послуг.

Прозорість як послуга – хмарна послуга з надання можливості відновлення інформації, що втрачається при переході до хмарних обчислень.

Службовий базовий інструментальний комплекс для розробки й супроводу програм як послуга – хмарна послуга з надання провайдером користувачеві можливості використання середовища створення, розгортання, виконання, керівництва й керування програмним забезпеченням, що реалізує одну або більше класів хмарних послуг.

«Все як послуга» – концепція, що полягає в наданні хмарним провайдером всіх можливих хмарних послуг користувачеві, що одержує право легітимного використання всього, що йому необхідно, при наявності тільки доступу в Інтернет.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.5. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

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

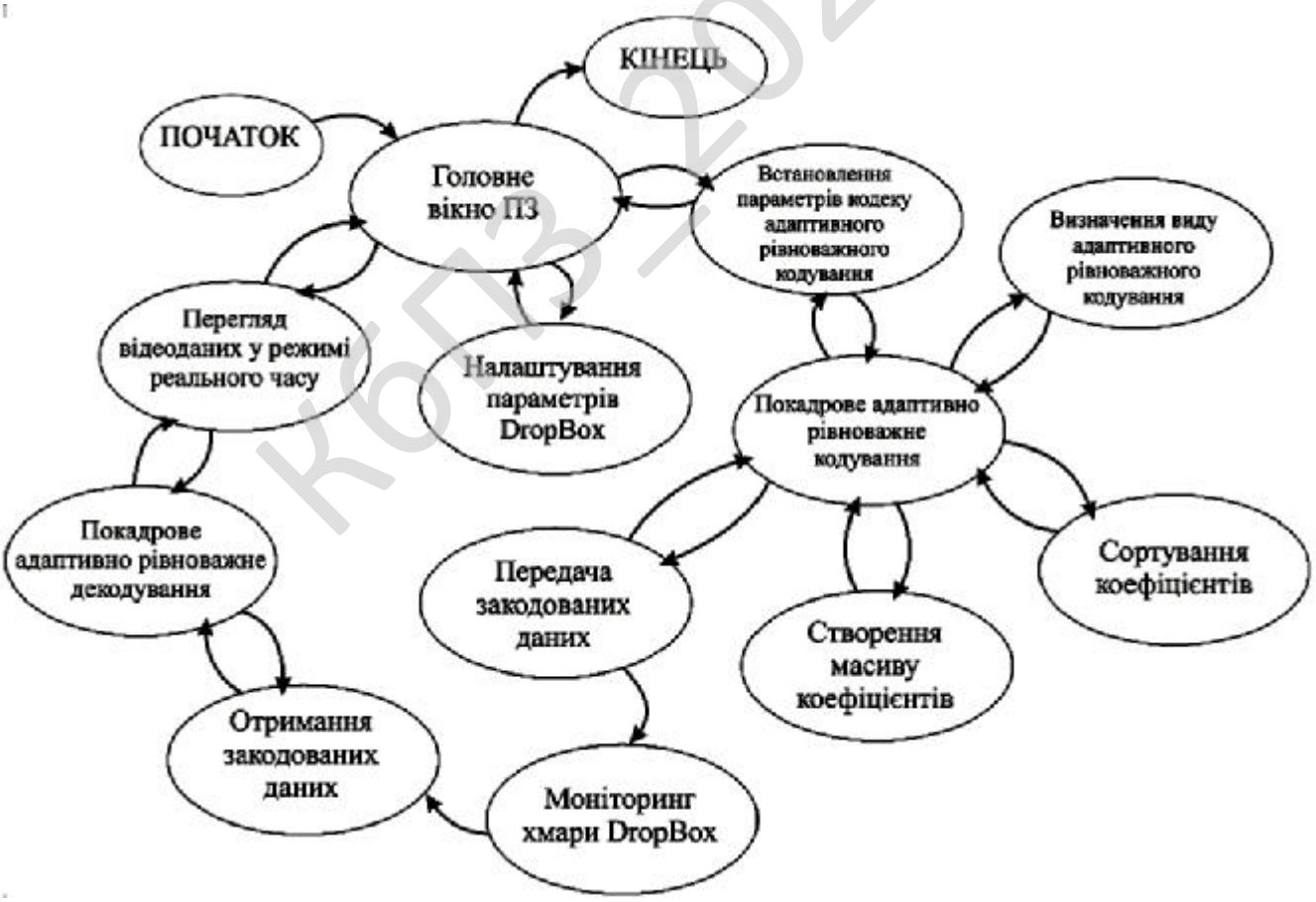

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

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

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограм та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограм виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Опис алгоритмів функціонування системи

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

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

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

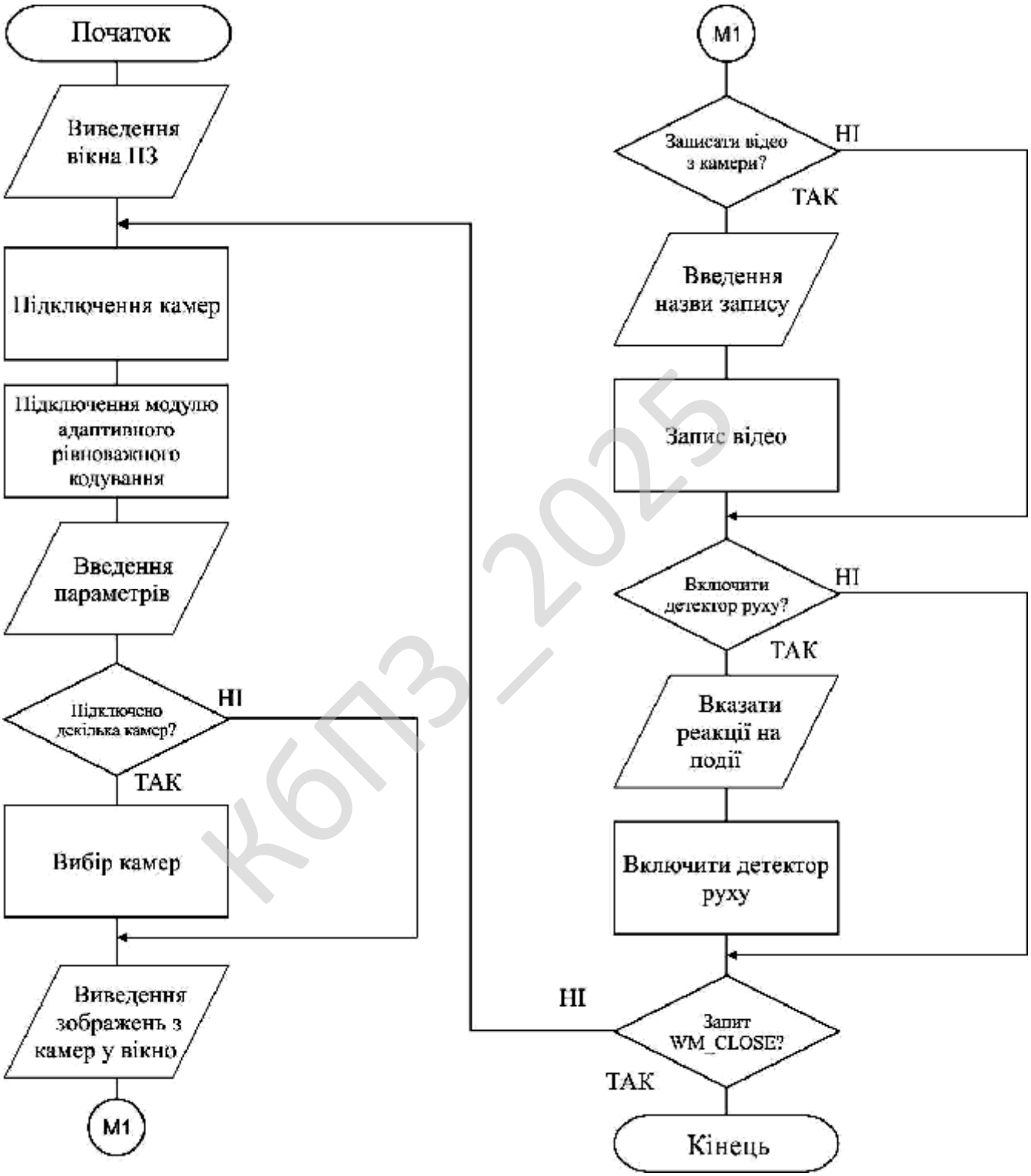

Рисунок 4.1 – Блок-схема основної програми

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

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

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

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

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

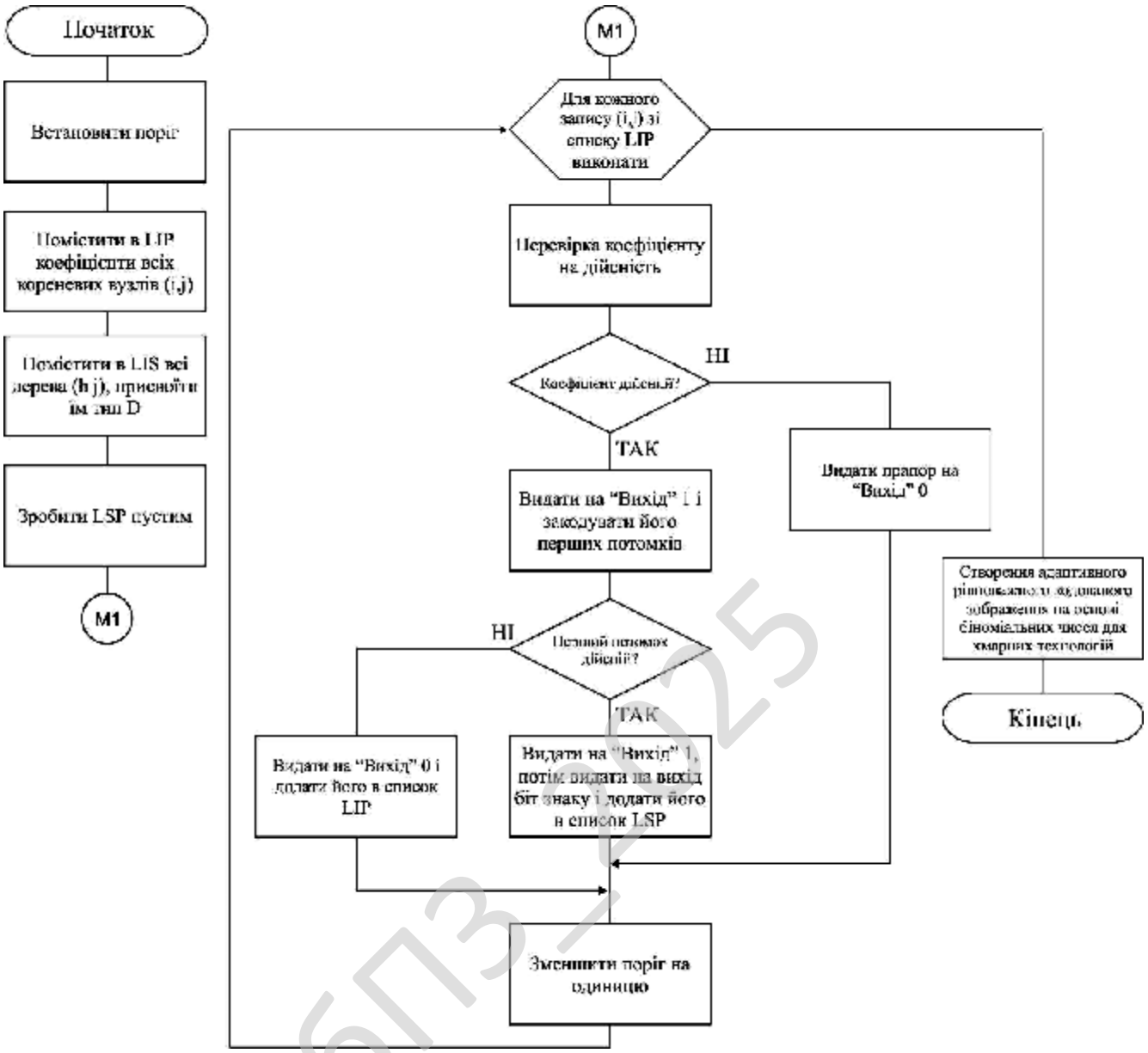

Рисунок 4.2 – Блок-схема роботи підпрограми

UML необхідний:

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

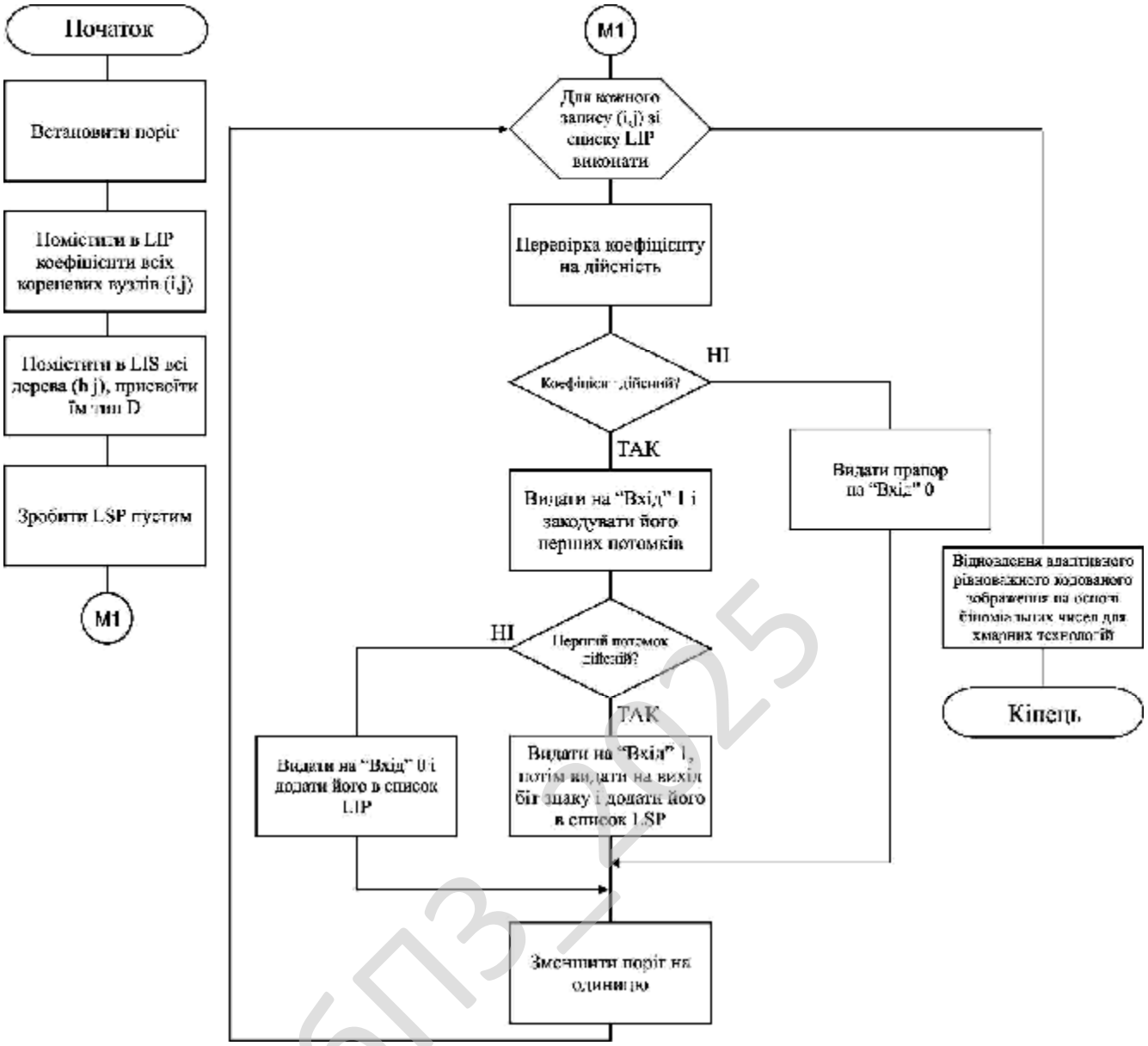

Рисунок 4.3 – Блок-схема роботи підпрограми

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

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

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності.

Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Нижче наведемо код алгоритму кодування SPIHT який використовується при створенні системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

```
namespace SPIHT
{
     using System;
     using System.Drawing;
     using System.IO;
     using System.Threading;
     using System.Net;
     using videosource;
     /// <summary>
/// SPIHTSource - Алгоритм SPIHT
     /// </summary>
     public class SPIHTSource : IVideoSource
     {
          private stringsource;
          private stringlogin = null;
          private stringpassword = null;
          private objectuserData = null;
          private int      framesReceived;
          private int      bytesReceived;
          private bool  useSeparateConnectionGroup = false;
          private bool  preventCaching = false;
          private int       frameInterval = 0;
// інтервал подання фреймів у мілісекундах
          private const int  bufSize = 512 * 1024;
// розмір буферу
          private const int  readSize = 1024;
// розмір блоку для читання
          private Threadthread = null;
          private ManualResetEvent stopEvent = null;
// нова подія у фреймі
          public event CameraEventHandler NewFrame;
          // SeparateConnectioGroup властивості
// індикатор відкриття WebRequest у групі
```

```
        public bool   SeparateConnectionGroup
        {
            get { return useSeparateConnectionGroup; }
            set { useSeparateConnectionGroup = value; }
        }
        // PreventCaching властивості
// Якщо властивості є правильними, то керуємо параметриу URL.
// Цей клієнт повинен бути встановлений на проксі-сервері.
        public bool   PreventCaching
        {
            get { return preventCaching; }
            set { preventCaching = value; }
        }
// FrameInterval властивості - інтервал між фреймами
//Якщо властивості встановлені в 100, тоді джерело формує 10 фреймів
// в секунду
        public int FrameInterval
        {
            get { return frameInterval; }
            set { frameInterval = value; }
        }
// VideoSource властивості
        public virtual string VideoSource
        {
            get { return source; }
            set { source = value; }
        }
// Властивості підключення
        public string Login
        {
            get { return login; }
            set { login = value; }
        }
// Властивості паролювання
        public string Password
        {
            get { return password; }
            set { password = value; }
        }
// FramesReceived властивості
        public int FramesReceived
        {
```

```
            get
            {
                int frames = framesReceived;
                framesReceived = 0;
                return frames;
            }
        }
// BytesReceived властивості
        public int BytesReceived
        {
            get
            {
                int bytes = bytesReceived;
                bytesReceived = 0;
                return bytes;
            }
        }
// UserData властивості
        public object UserData
        {
            get { return userData; }
            set { userData = value; }
        }
// Отримуємо стан вихідного відео
        public bool Running
        {
            get
            {
                if (thread != null)
                {
                    if (thread.Join(0) == false)
                        return true;
// Якщо стан не заданий, звільнюємо ресурси
                    Free();
                }
                return false;
            }
        }
// Конструктор
        public SPIHTSource()
        {
        }
```

```
// Починаємо роботу
          public void Start()
          {
               if (thread == null)
               {
                    framesReceived = 0;
                    bytesReceived = 0;
// Створюємо подію
                    stopEvent = new ManualResetEvent(false);

// Створюємо й стартуємо нову подію
                    thread = new Thread(new ThreadStart(WorkerThread));
                    thread.Name = source;
                    thread.Start();
               }
          }
// Сигнал події до остановки роботи
          public void SignalToStop()
          {
// Остановлюємо подію
               if (thread != null)
               {
// Сигнал остановки
                    stopEvent.Set();
               }
          }
          // Чекаємо остановки події
          public void WaitForStop()
          {
               if (thread != null)
               {
// Чекаємо остановки події
                    thread.Join();
                    Free();
               }
          }
// Подія помилки
          public void Stop()
          {
               if (this.Running)
               {
                    thread.Abort();
```

```
                    WaitForStop();
               }
          }
// Визволяємо ресурси
          private void Free()
          {
               thread = null;
               // Випуск події
               stopEvent.Close();
               stopEvent = null;
          }
// Точка входу події
          public void WorkerThread()
          {
               byte[] buffer = new byte[bufSize];
// буфер читання потоку
               HttpWebRequestreq = null;
               WebResponse        resp = null;
               Stream             stream = null;
               Random             rnd = new Random((int) DateTime.Now.Ticks);
               DateTime      start;
               TimeSpan      span;
               while (true)
               {
                    int  read, total = 0;
                    try
                    {
                         start = DateTime.Now;
// створюємо запрос
                         if (!preventCaching)
                         {
                              req = (HttpWebRequest) WebRequest.Create(source);
                         }
                         else
                         {
                    req = (HttpWebRequest) WebRequest.Create(source +
((source.IndexOf('?') == -1) ? '?' : '&') + "fake=" + rnd.Next().ToString());
                         }
// встановлюємо логін та пароль
                    if ((login != null) && (password != null) && (login != ""))
                         req.Credentials = new NetworkCredential(login, password);
// встановлюємо найменування групи підключення
```

```
                        if (useSeparateConnectionGroup)
                             req.ConnectionGroupName = GetHashCode().ToString();
// отримуємо відповідь
                        resp = req.GetResponse();
// отримуємо відповідь потоку
                        stream = resp.GetResponseStream();
// цикл
                        while (!stopEvent.WaitOne(0, true))
                        {
// перевіряємо загальне читання
                             if (total > bufSize - readSize)
                             { total = 0; }
// Читаємо наступний блок у потоці
                        if ((read = stream.Read(buffer, total, readSize)) == 0)
                                  break;
                             total += read;
// Додаємо лічильник зчитаних байт
                             bytesReceived += read;
                        }
                        if (!stopEvent.WaitOne(0, true)) {
// додаємо лічильник фреймів
                             framesReceived++;
// остановка читання зображення
                             if (NewFrame != null) {
Bitmap    bmp = (Bitmap) Bitmap.FromStream(new MemoryStream(buffer, 0, total));
// Клієнт
                                  NewFrame(this, new CameraEventArgs(bmp));
// Будуємо картинку
                                  bmp.Dispose();
                                  bmp = null;
                             }
                        }
// Чекаємо в циклі ?
                        if (frameInterval > 0)
                        {
// діапазон часу
                             span = DateTime.Now.Subtract(start);
// засипання у мілісекундах
int msec = frameInterval - (int) span.TotalMilliseconds;
               while ((msec > 0) && (stopEvent.WaitOne(0, true) == false))
                        {
// засипання ...
```

```
                                Thread.Sleep((msec < 100) ? msec : 100);
                                msec -= 100;
                            }
                        }
                    }
                    catch (WebException ex)
                    {
          System.Diagnostics.Debug.WriteLine("=============: " + ex.Message);
// чекаємо у циклі наступну спробу
                        Thread.Sleep(250);
                    }
                    catch (Exception ex)
                    {
               System.Diagnostics.Debug.WriteLine("=============: " + ex.Message);
                    }
                    finally
                    {
// помилка запиту
                        if (req != null)
                        {
                            req.Abort();
                            req = null;
                        }
// закриваємо потоку
                        if (stream != null)
                        {
                            stream.Close();
                            stream = null;
                        }
// close response
                        if (resp != null)
                        {
                            resp.Close();
                            resp = null;
                        }
                    }
// need to stop ?
                    if (stopEvent.WaitOne(0, true))
                        break;
                }
            }
        }
```

}

Таким чином розглянувши блок–схеми та опис алгоритмів функціонування системи, перейдемо до опису алгоритму та механізмів захисту розробленого програмного забезпечення.

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою наступного алгоритму захисту інформації DSA.

Відправник і одержувач електронного документа використовують при обчисленні великі цілі числа: G і P – прості числа, L біт кожне ($512 < L < 1024$); q – простої число довжиною 160 біт (дільник числа (P-1). Числа G, P, q є відкритими й можуть бути загальними для всіх користувачів мережі.

Відправник вибирає випадкове ціле число X, $1 < X < q$. Число X є секретним ключем відправника для формування електронного цифрового підпису. Потім відправник обчислює значення $Y = G^X \bmod P$. Число Y є відкритим ключем для перевірки підпису відправника. Число Y передається всім одержувачам документів.

Цей алгоритм також передбачає використання однобічної функції гешування h(-). У стандарті DSS визначений алгоритм безпечного гешування SHA. Для того щоб підписати документ M, відправник гешує його в ціле геш-значення m: $m = h(M)$, $1 < m < q$, потім генерує випадкове ціле число K, $1 < K < q$, і обчислює число r: $r = (G^K \bmod P) \bmod q$. Потім відправник обчислює за допомогою секретного ключа X ціле число s:

$$s = \frac{m + r * X}{K} \bmod q \ .$$

Пара чисел r і s утворить цифровий підпис $S = (r, s)$ під документом M. Таким чином, підписане повідомлення являє собою трійку чисел $[M,r,s]$. Одержувач підписаного повідомлення $[M, r, s]$ перевіряє виконання умов $0 < r < q$, $0 < s < q$ і відкидає підпис, якщо хоча б одна із цих умов не виконана.

Потім одержувач обчислює значення $w=1/s \bmod q$, геш-значення $m = h(M)$ і числа $u_1 = (m*w) \bmod q$, $u_2 = (r*w) \bmod q$. Далі одержувач за допомогою відкритого ключа Y обчислює значення $v = ((G^{U1} * Y^{U2}) \bmod P) \bmod q$ і перевіряє виконання умови $v = r$. Якщо умова $v = r$ виконується, тоді підпис $S = (r,s)$ під документом M визнається одержувачем справжнім.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

Розроблене програмне забезпечення адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Камери; Відеозаписи; Детектор руху; Параметри; Довідка.

– Функції представлені у графічному вигляді (іконки).

– Вікна обрання камери.

– Навігаційного меню яке визивається натисканням правої клавіші маніпулятора миші.

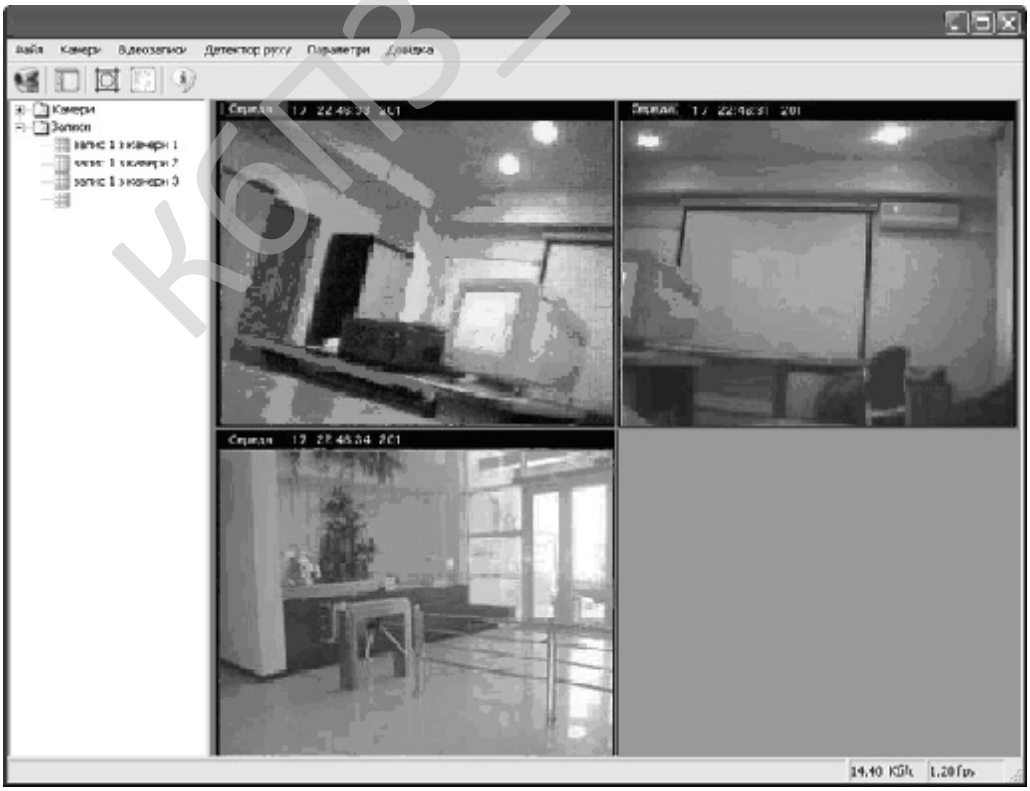

Рисунок 5.1 – Головне вікно розробленого ПЗ

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

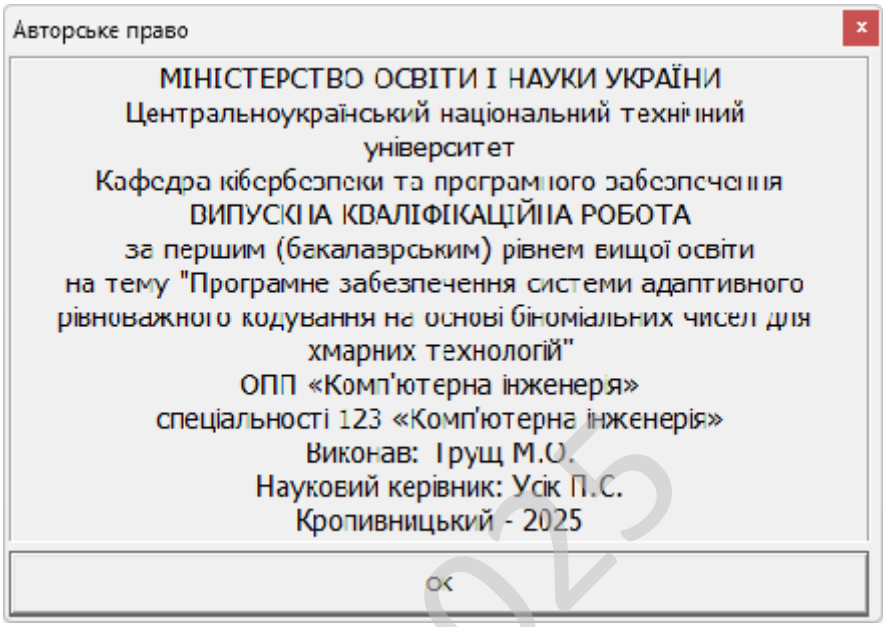

Рисунок 5.2 – Вікно розробника ПЗ

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

Проводилось тестування форматом білої скриньки.

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

– Досліджена система адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

– На основі отриманих результатів досліджень створена програмна реалізація системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

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

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

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

системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм DSA.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

2. 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.

3. 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.

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

5. 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.

6. 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.

7. 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.

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

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. Smirnov O. Kuznetsov A., Zaicenko 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. Smirnov A.A., Kuzne tsov 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.

25. Смірнова Т.В., Конопліцька-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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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

33. 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.

34. 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.

35. Смірнов О.А., Конопліцька-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.

36. 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.

37. Смірнов О.А., Дрєєва Г.М., Дрєєв 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.

38. Смірнов О.А., Конопліцька-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.

39. 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).

40. Смірнов О.А., Дрєєва 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

41. Дрєєва Г.М., Смірнов О.А., Дрєєв 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.

42. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв 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.

43. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-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.

44. 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.

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

46. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв 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.

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

48. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв 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.

49. 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.

50. 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.

51. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 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 адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Visual C#.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 65 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).

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.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Усік П.С.

Програмне забезпечення системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 41

Літера: РП

Кропивницький – 2025 року

Файл MultimodeNetCloud.cs – робота з хмарами у мережі

```
namespace multisource
{
	using System;
	using System.Collections;
	using NetCloud;
	using BinomialCode;
	using mBinomialCode;

	/// <summary>
	/// MultimodeNetCloud - абстрактний клас для роботи з хмарами, з
підтримкою  /// мульти робочого режиму
	/// </summary>
	public abstract class MultimodeNetCloud : INetCloud
	{
		protected INetCloud	NetCloud;
		protected StreamType	streamType;
		private ArrayList	delegates = new ArrayList();

		// Нова подія у фреймі
		public event CloudEventHandler NewFrame
		{
			add
			{
				NetCloud.NewFrame += value;
				delegates.Add((object) value);
			}
			remove
			{
				NetCloud.NewFrame -= value;
				delegates.Remove((object) value);
			}
		}

		// Конструктор
		public MultimodeNetCloud()
		{
		}

		// StreamType властивості
		public virtual StreamType StreamType
		{
			get { return streamType; }
			set
			{
				// покращує потоковий тип,
				if ((streamType != value) && (!NetCloud.Running))
				{
					streamType = value;

					// зберігає дані
					object	userData = NetCloud.UserData;
					string	login = NetCloud.Login;
					string	password = NetCloud.Password;

					// створює нове базове хмарне середовище у мережі
					switch (streamType)
					{
						case StreamType.BinomialCode:
							NetCloud = new BinomialCodeSource();
							break;
						case StreamType.MBinomialCode:
							NetCloud = new MBinomialCodeSource();
							break;
					}
```

```
                                    // бере дані та повертає у нове хмарне середовище
у мережі
                                    NetCloud.Login          = login;
                                    NetCloud.Password = password;
                                    NetCloud.UserData = userData;

                                    // додає делегування до NewFrame події
                                    foreach (object handler in delegates)
                                          NetCloud.NewFrame += (CloudEventHandler)
handler;

                                    UpdateNetCloud();
                              }
                        }
                  }

                  // Властивості хмарного середовища у мережі
                  public abstract string NetCloud
                  {
                        get;
                        set;
                  }

                  // Властивості підключення
                  public string Login
                  {
                        get { return NetCloud.Login; }
                        set { NetCloud.Login = value; }
                  }

                  // Властивості паролювання
                  public string Password
                  {
                        get { return NetCloud.Password; }
                        set { NetCloud.Password = value; }
                  }

                  // FramesReceived властивості
                  public int FramesReceived
                  {
                        get { return NetCloud.FramesReceived; }
                  }

                  // BytesReceived властивості
                  public int BytesReceived
                  {
                        get { return NetCloud.BytesReceived; }
                  }

                  // UserData властивості
                  public object UserData
                  {
                        get { return NetCloud.UserData; }
                        set { NetCloud.UserData = value; }
                  }

                  // Беремо стан хмарного середовища у мережі
                  public bool Running
                  {
                        get { return NetCloud.Running; }
                  }

                  // Починаємо отримувати дані з хмарного середовища у мережі
                  public void Start()
                  {
                        NetCloud.Start();
                  }
```

```
            // Закінчуємо отримувати дані з хмарного середовища у мережі
            public void SignalToStop()
            {
                  NetCloud.SignalToStop();
            }

            // Чекаємо закінчення
            public void WaitForStop()
            {
                  NetCloud.WaitForStop();
            }

            // Закінчення роботи
            public void Stop()
            {
                  NetCloud.Stop();
            }

            // Обновлюємо хмарне середовище у мережі
            protected abstract void UpdateNetCloud();
      }
}
```

Файл CloudInfo.cs – отримання інформації про хмару

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace CloudViewer
{
	/// <summary>
	/// інформація з CloudInfo.
	/// </summary>
	public class CloudInfo : System.Windows.Forms.Form
	{
		private Cloud Cloud;

		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.Label widthLabel;
		private System.Windows.Forms.Label label2;
		private System.Windows.Forms.Label label3;
		private System.Windows.Forms.Label nameLabel;
		private System.Windows.Forms.Label label5;
		private System.Windows.Forms.Label descriptionLabel;
		private System.Windows.Forms.Label label4;
		private System.Windows.Forms.Label providerLabel;
		private System.Windows.Forms.PictureBox pictureBox1;
		private System.Windows.Forms.Label heightLabel;
		private System.Windows.Forms.Button closeButton;
		private System.Windows.Forms.PictureBox pictureBox2;
		/// <summary>
		/// Необхідні змінні розробника.
		/// </summary>
		private System.ComponentModel.Container components = null;

		// Cloud властивості
		public Cloud Cloud
		{
			get { return Cloud; }
			set { Cloud = value; }
		}

		// Конструктор
		public CloudInfo()
		{
			//
			// Необхідно для підтримки Windows Form Designer
			//
			InitializeComponent();

			//
			// ПРИМІТКА Додаємо любий конструктор коду після виклику
InitializeComponent
			//
		}

		/// <summary>
		/// Очищуємо використані ресурси.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
```

```
            }
            base.Dispose( disposing );
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Необхідний метод розробника – не модифікується
        /// зміст цього методу з редактором коду.
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.widthLabel = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.nameLabel = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.descriptionLabel = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.providerLabel = new System.Windows.Forms.Label();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.heightLabel = new System.Windows.Forms.Label();
            this.closeButton = new System.Windows.Forms.Button();
            this.pictureBox2 = new System.Windows.Forms.PictureBox();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(8, 136);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(40, 14);
            this.label1.TabIndex = 0;
            this.label1.Text = "Width:";
            //
            // widthLabel
            //
            this.widthLabel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.widthLabel.Location = new System.Drawing.Point(60, 135);
            this.widthLabel.Name = "widthLabel";
            this.widthLabel.Size = new System.Drawing.Size(60, 16);
            this.widthLabel.TabIndex = 1;
            this.widthLabel.TextAlign =
System.Drawing.ContentAlignment.TopCenter;
            //
            // label2
            //
            this.label2.Location = new System.Drawing.Point(140, 136);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(40, 14);
            this.label2.TabIndex = 2;
            this.label2.Text = "Height:";
            //
            // label3
            //
            this.label3.Location = new System.Drawing.Point(8, 9);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(40, 14);
            this.label3.TabIndex = 3;
            this.label3.Text = "Name:";
            //
            // nameLabel
            //
            this.nameLabel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.nameLabel.Location = new System.Drawing.Point(60, 8);
            this.nameLabel.Name = "nameLabel";
            this.nameLabel.Size = new System.Drawing.Size(200, 16);
            this.nameLabel.TabIndex = 4;
```

```
                    //
                    // label5
                    //
                    this.label5.Location = new System.Drawing.Point(8, 30);
                    this.label5.Name = "label5";
                    this.label5.Size = new System.Drawing.Size(65, 14);
                    this.label5.TabIndex = 5;
                    this.label5.Text = "Description:";
                    //
                    // descriptionLabel
                    //
                    this.descriptionLabel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
                    this.descriptionLabel.Location = new System.Drawing.Point(8,
47);
                    this.descriptionLabel.Name = "descriptionLabel";
                    this.descriptionLabel.Size = new System.Drawing.Size(252, 40);
                    this.descriptionLabel.TabIndex = 6;
                    //
                    // label4
                    //
                    this.label4.Location = new System.Drawing.Point(8, 96);
                    this.label4.Name = "label4";
                    this.label4.Size = new System.Drawing.Size(50, 14);
                    this.label4.TabIndex = 7;
                    this.label4.Text = "Provider:";
                    //
                    // providerLabel
                    //
                    this.providerLabel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
                    this.providerLabel.Location = new System.Drawing.Point(60,
95);
                    this.providerLabel.Name = "providerLabel";
                    this.providerLabel.Size = new System.Drawing.Size(200, 16);
                    this.providerLabel.TabIndex = 8;
                    //
                    // pictureBox1
                    //
                    this.pictureBox1.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
                    this.pictureBox1.Location = new System.Drawing.Point(8, 120);
                    this.pictureBox1.Name = "pictureBox1";
                    this.pictureBox1.Size = new System.Drawing.Size(252, 2);
                    this.pictureBox1.TabIndex = 9;
                    this.pictureBox1.TabStop = false;
                    //
                    // heightLabel
                    //
                    this.heightLabel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
                    this.heightLabel.Location = new System.Drawing.Point(200,
135);
                    this.heightLabel.Name = "heightLabel";
                    this.heightLabel.Size = new System.Drawing.Size(60, 16);
                    this.heightLabel.TabIndex = 10;
                    this.heightLabel.TextAlign =
System.Drawing.ContentAlignment.TopCenter;
                    //
                    // closeButton
                    //
                    this.closeButton.DialogResult =
System.Windows.Forms.DialogResult.OK;
                    this.closeButton.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
                    this.closeButton.Location = new System.Drawing.Point(98, 175);
                    this.closeButton.Name = "closeButton";
                    this.closeButton.TabIndex = 11;
                    this.closeButton.Text = "Close";
```

```
                //
                // pictureBox2
                //
                this.pictureBox2.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
                this.pictureBox2.Location = new System.Drawing.Point(9, 163);
                this.pictureBox2.Name = "pictureBox2";
                this.pictureBox2.Size = new System.Drawing.Size(252, 2);
                this.pictureBox2.TabIndex = 12;
                this.pictureBox2.TabStop = false;
                //
                // CloudInfo
                //
                this.AcceptButton = this.closeButton;
                this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
                this.CancelButton = this.closeButton;
                this.ClientSize = new System.Drawing.Size(270, 206);
                this.Controls.AddRange(new System.Windows.Forms.Control[] {

                                this.pictureBox2,

                                this.closeButton,

                                this.heightLabel,

                                this.pictureBox1,

                                this.providerLabel,

                                this.label4,

                                this.descriptionLabel,

                                this.label5,

                                this.nameLabel,

                                this.label3,

                                this.label2,

                                this.widthLabel,

                                this.label1});
                this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedToolWindow;
                this.MaximizeBox = false;
                this.MinimizeBox = false;
                this.Name = "CloudInfo";
                this.Opacity = 0.85;
                this.ShowInTaskbar = false;
                this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterParent;
                this.Text = "Cloud info";
                this.Load += new System.EventHandler(this.CloudInfo_Load);
                this.ResumeLayout(false);

            }
            #endregion

            // On load
            private void CloudInfo_Load(object sender, System.EventArgs e)
            {
                  if (Cloud != null)
                  {
                        nameLabel.Text = Cloud.Name;
                        descriptionLabel.Text = Cloud.Description;
                        providerLabel.Text = Cloud.Provider.Name;
```

```
                        if (Cloud.Width != -1)
                        {
                              widthLabel.Text = Cloud.Width.ToString();
                              heightLabel.Text = Cloud.Height.ToString();
                        }
                        else
                        {
                              widthLabel.Text = string.Empty;
                              heightLabel.Text = string.Empty;
                        }
                  }
            }
      }
}
```

Файл Multiplexer.cs – мультиплексування даних (для роботи з декількома хмарами)

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;

namespace CloudViewer
{
      /// <summary>
      /// Загальний опис для мультиплексора.
      /// </summary>
      public class Multiplexer : System.Windows.Forms.Panel
      {
            private const int       MaxRows = 5;
            private const int       MaxCols = 5;
            private CloudWindow[,]  camWindows;

            private bool      fitToWindow = false;
            private bool      singleCloudMode = true;
            private bool      CloudsVisible = false;

            private int       rows = 1;
            private int       cols = 1;
            private int       cellWidth = 320;
            private int       cellHeight = 240;

            private CloudWindow     lastClicked;

            private CloudViewer.CloudWindow CloudWindow1;
            private CloudViewer.CloudWindow CloudWindow2;
            private CloudViewer.CloudWindow CloudWindow3;
            private CloudViewer.CloudWindow CloudWindow4;
            private CloudViewer.CloudWindow CloudWindow5;
            private CloudViewer.CloudWindow CloudWindow6;
            private CloudViewer.CloudWindow CloudWindow7;
            private CloudViewer.CloudWindow CloudWindow8;
            private CloudViewer.CloudWindow CloudWindow9;
            private CloudViewer.CloudWindow CloudWindow10;
            private CloudViewer.CloudWindow CloudWindow11;
            private CloudViewer.CloudWindow CloudWindow12;
            private CloudViewer.CloudWindow CloudWindow13;
            private CloudViewer.CloudWindow CloudWindow14;
            private CloudViewer.CloudWindow CloudWindow15;
            private CloudViewer.CloudWindow CloudWindow16;
            private CloudViewer.CloudWindow CloudWindow17;
            private CloudViewer.CloudWindow CloudWindow18;
            private CloudViewer.CloudWindow CloudWindow19;
            private CloudViewer.CloudWindow CloudWindow20;
            private CloudViewer.CloudWindow CloudWindow21;
            private CloudViewer.CloudWindow CloudWindow22;
            private CloudViewer.CloudWindow CloudWindow23;
            private CloudViewer.CloudWindow CloudWindow24;
            private CloudViewer.CloudWindow CloudWindow25;
            /// <summary>
            /// Опис змінних розробника.
            /// </summary>
            private System.ComponentModel.Container components = null;

            // FitToWindow властивості
            [DefaultValue(false)]
            public bool FitToWindow
            {
                  get { return fitToWindow; }
                  set
```

```
                {
                        fitToWindow = value;

                        if ((camWindows[0, 0].AutoSize = (!fitToWindow &&
singleCloudMode)) == true)
                        {
                                camWindows[0, 0].UpdatePosition();
                        }
                        else
                        {
                                UpdateSize();
                        }
                }
        }
        // SingleCloudMode властивості
        [DefaultValue(true)]
        public bool SingleCloudMode
        {
                get { return singleCloudMode; }
                set
                {
                        singleCloudMode = value;
                        if (!fitToWindow)
                                camWindows[0, 0].AutoSize = value;
                }
        }
        // CloudsVisible властивості
        [DefaultValue(false)]
        public bool CloudsVisible
        {
                get { return CloudsVisible; }
                set
                {
                        CloudsVisible = value;

                        // Показувати/приховувати усі хмари
                        for (int i = 0; i < rows; i++)
                        {
                                for (int j = 0; j < cols; j++)
                                {
                                        camWindows[i, j].Visible = value;
                                }
                        }
                }
        }
        // Rows властивості
        [DefaultValue(1)]
        public int Rows
        {
                get { return rows; }
                set
                {
                        rows = Math.Max(1, Math.Min(MaxRows, value));
                        UpdateVisiblity();
                        UpdateSize();
                }
        }
        // Cols властивості
        [DefaultValue(1)]
        public int Cols
        {
                get { return cols; }
                set
                {
                        cols = Math.Max(1, Math.Min(MaxCols, value));
                        UpdateVisiblity();
                        UpdateSize();
                }
        }
```

```
		// CellWidth
		[DefaultValue(320)]
		public int CellWidth
		{
			get { return cellWidth; }
			set
			{
				cellWidth = Math.Max(50, Math.Min(800, value));
				UpdateSize();
			}
		}
		// CellHeight
		[DefaultValue(240)]
		public int CellHeight
		{
			get { return cellHeight; }
			set
			{
				cellHeight = Math.Max(50, Math.Min(800, value));
				UpdateSize();
			}
		}
		// Контекстне меню у вікні хмари
		[DefaultValue(null)]
		public ContextMenu CloudsContextMenu
		{
			get { return camWindows[0, 0].ContextMenu; }
			set
			{
				for (int i = 0; i < MaxRows; i++)
				{
					for (int j = 0; j < MaxCols; j++)
					{
						camWindows[i, j].ContextMenu = value;
					}
				}
			}
		}
		// Хмара при останньому нажатті
		[Browsable(false)]
		public Cloud ContextCloud
		{
			get { return (lastClicked == null) ? null : lastClicked.Cloud;
}
		}

		// Конструктор
		public Multiplexer()
		{
			// Цей виклик використовується у Windows.Forms Form Designer.
			InitializeComponent();

			// ПРИМІТКА Додається ініціалізація після виклику InitForm
			camWindows = new CloudWindow[MaxRows, MaxCols];

			// row 1
			camWindows[0, 0] = CloudWindow1;
			camWindows[0, 1] = CloudWindow2;
			camWindows[0, 2] = CloudWindow3;
			camWindows[0, 3] = CloudWindow4;
			camWindows[0, 4] = CloudWindow5;
			// row 2
			camWindows[1, 0] = CloudWindow6;
			camWindows[1, 1] = CloudWindow7;
			camWindows[1, 2] = CloudWindow8;
			camWindows[1, 3] = CloudWindow9;
			camWindows[1, 4] = CloudWindow10;
			// row 3
			camWindows[2, 0] = CloudWindow11;
```

```
            camWindows[2, 1] = CloudWindow12;
            camWindows[2, 2] = CloudWindow13;
            camWindows[2, 3] = CloudWindow14;
            camWindows[2, 4] = CloudWindow15;
            // row 4
            camWindows[3, 0] = CloudWindow16;
            camWindows[3, 1] = CloudWindow17;
            camWindows[3, 2] = CloudWindow18;
            camWindows[3, 3] = CloudWindow19;
            camWindows[3, 4] = CloudWindow20;
            // row 5
            camWindows[4, 0] = CloudWindow21;
            camWindows[4, 1] = CloudWindow22;
            camWindows[4, 2] = CloudWindow23;
            camWindows[4, 3] = CloudWindow24;
            camWindows[4, 4] = CloudWindow25;
      }

      /// <summary>
      /// Очищуємо усі ресурси використовувані користувачем.
      /// </summary>
      protected override void Dispose( bool disposing )
      {
            if( disposing )
            {
                  if(components != null)
                  {
                        components.Dispose();
                  }
            }
            base.Dispose( disposing );
      }

      #region Component Designer generated code
      /// <summary>
      /// Необхідний метод для підтримки розробника - не модифікується
      /// контент цього методу з редактором коду.
      /// </summary>
      private void InitializeComponent()
      {
            this.CloudWindow1 = new CloudViewer.CloudWindow();
            this.CloudWindow2 = new CloudViewer.CloudWindow();
            this.CloudWindow3 = new CloudViewer.CloudWindow();
            this.CloudWindow4 = new CloudViewer.CloudWindow();
            this.CloudWindow5 = new CloudViewer.CloudWindow();
            this.CloudWindow6 = new CloudViewer.CloudWindow();
            this.CloudWindow7 = new CloudViewer.CloudWindow();
            this.CloudWindow8 = new CloudViewer.CloudWindow();
            this.CloudWindow9 = new CloudViewer.CloudWindow();
            this.CloudWindow10 = new CloudViewer.CloudWindow();
            this.CloudWindow11 = new CloudViewer.CloudWindow();
            this.CloudWindow12 = new CloudViewer.CloudWindow();
            this.CloudWindow13 = new CloudViewer.CloudWindow();
            this.CloudWindow14 = new CloudViewer.CloudWindow();
            this.CloudWindow15 = new CloudViewer.CloudWindow();
            this.CloudWindow16 = new CloudViewer.CloudWindow();
            this.CloudWindow17 = new CloudViewer.CloudWindow();
            this.CloudWindow18 = new CloudViewer.CloudWindow();
            this.CloudWindow19 = new CloudViewer.CloudWindow();
            this.CloudWindow20 = new CloudViewer.CloudWindow();
            this.CloudWindow21 = new CloudViewer.CloudWindow();
            this.CloudWindow22 = new CloudViewer.CloudWindow();
            this.CloudWindow23 = new CloudViewer.CloudWindow();
            this.CloudWindow24 = new CloudViewer.CloudWindow();
            this.CloudWindow25 = new CloudViewer.CloudWindow();
            this.SuspendLayout();
            //
            // CloudWindow1
            //
```

```
                this.CloudWindow1.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow1.Cloud = null;
                this.CloudWindow1.Location = new System.Drawing.Point(285,
17);
                this.CloudWindow1.Name = "CloudWindow1";
                this.CloudWindow1.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow1.TabIndex = 0;
                this.CloudWindow1.Text = "CloudWindow1";
                this.CloudWindow1.Visible = false;
                this.CloudWindow1.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow2
                //
                this.CloudWindow2.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow2.Cloud = null;
                this.CloudWindow2.Location = new System.Drawing.Point(151,
17);
                this.CloudWindow2.Name = "CloudWindow2";
                this.CloudWindow2.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow2.TabIndex = 1;
                this.CloudWindow2.Text = "CloudWindow2";
                this.CloudWindow2.Visible = false;
                this.CloudWindow2.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow3
                //
                this.CloudWindow3.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow3.Cloud = null;
                this.CloudWindow3.Location = new System.Drawing.Point(419,
17);
                this.CloudWindow3.Name = "CloudWindow3";
                this.CloudWindow3.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow3.TabIndex = 2;
                this.CloudWindow3.Text = "CloudWindow3";
                this.CloudWindow3.Visible = false;
                this.CloudWindow3.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow4
                //
                this.CloudWindow4.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow4.Cloud = null;
                this.CloudWindow4.Location = new System.Drawing.Point(553,
17);
                this.CloudWindow4.Name = "CloudWindow4";
                this.CloudWindow4.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow4.TabIndex = 3;
                this.CloudWindow4.Text = "CloudWindow4";
                this.CloudWindow4.Visible = false;
                this.CloudWindow4.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow5
                //
                this.CloudWindow5.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow5.Cloud = null;
                this.CloudWindow5.Location = new System.Drawing.Point(17, 54);
                this.CloudWindow5.Name = "CloudWindow5";
                this.CloudWindow5.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow5.TabIndex = 4;
                this.CloudWindow5.Text = "CloudWindow5";
                this.CloudWindow5.Visible = false;
```

```
                this.CloudWindow5.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow6
                //
                this.CloudWindow6.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow6.Cloud = null;
                this.CloudWindow6.Location = new System.Drawing.Point(17, 91);
                this.CloudWindow6.Name = "CloudWindow6";
                this.CloudWindow6.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow6.TabIndex = 5;
                this.CloudWindow6.Text = "CloudWindow6";
                this.CloudWindow6.Visible = false;
                this.CloudWindow6.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow7
                //
                this.CloudWindow7.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow7.Cloud = null;
                this.CloudWindow7.Location = new System.Drawing.Point(17, 91);
                this.CloudWindow7.Name = "CloudWindow7";
                this.CloudWindow7.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow7.TabIndex = 6;
                this.CloudWindow7.Text = "CloudWindow7";
                this.CloudWindow7.Visible = false;
                this.CloudWindow7.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow8
                //
                this.CloudWindow8.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow8.Cloud = null;
                this.CloudWindow8.Location = new System.Drawing.Point(17, 91);
                this.CloudWindow8.Name = "CloudWindow8";
                this.CloudWindow8.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow8.TabIndex = 7;
                this.CloudWindow8.Text = "CloudWindow8";
                this.CloudWindow8.Visible = false;
                this.CloudWindow8.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow9
                //
                this.CloudWindow9.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow9.Cloud = null;
                this.CloudWindow9.Location = new System.Drawing.Point(151,
91);
                this.CloudWindow9.Name = "CloudWindow9";
                this.CloudWindow9.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow9.TabIndex = 8;
                this.CloudWindow9.Text = "CloudWindow9";
                this.CloudWindow9.Visible = false;
                this.CloudWindow9.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow10
                //
                this.CloudWindow10.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow10.Cloud = null;
                this.CloudWindow10.Location = new System.Drawing.Point(328,
80);
                this.CloudWindow10.Name = "CloudWindow10";
                this.CloudWindow10.Size = new System.Drawing.Size(75, 64);
```

```
                this.CloudWindow10.TabIndex = 9;
                this.CloudWindow10.Text = "CloudWindow10";
                this.CloudWindow10.Visible = false;
                this.CloudWindow10.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow11
                //
                this.CloudWindow11.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow11.Cloud = null;
                this.CloudWindow11.Location = new System.Drawing.Point(8,
152);
                this.CloudWindow11.Name = "CloudWindow11";
                this.CloudWindow11.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow11.TabIndex = 10;
                this.CloudWindow11.Text = "CloudWindow11";
                this.CloudWindow11.Visible = false;
                this.CloudWindow11.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow12
                //
                this.CloudWindow12.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow12.Cloud = null;
                this.CloudWindow12.Location = new System.Drawing.Point(88,
152);
                this.CloudWindow12.Name = "CloudWindow12";
                this.CloudWindow12.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow12.TabIndex = 11;
                this.CloudWindow12.Text = "CloudWindow12";
                this.CloudWindow12.Visible = false;
                this.CloudWindow12.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow13
                //
                this.CloudWindow13.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow13.Cloud = null;
                this.CloudWindow13.Location = new System.Drawing.Point(228,
152);
                this.CloudWindow13.Name = "CloudWindow13";
                this.CloudWindow13.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow13.TabIndex = 12;
                this.CloudWindow13.Text = "CloudWindow13";
                this.CloudWindow13.Visible = false;
                this.CloudWindow13.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow14
                //
                this.CloudWindow14.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow14.Cloud = null;
                this.CloudWindow14.Location = new System.Drawing.Point(248,
152);
                this.CloudWindow14.Name = "CloudWindow14";
                this.CloudWindow14.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow14.TabIndex = 13;
                this.CloudWindow14.Text = "CloudWindow14";
                this.CloudWindow14.Visible = false;
                this.CloudWindow14.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow15
                //
```

```
                this.CloudWindow15.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow15.Cloud = null;
                this.CloudWindow15.Location = new System.Drawing.Point(388,
152);
                this.CloudWindow15.Name = "CloudWindow15";
                this.CloudWindow15.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow15.TabIndex = 14;
                this.CloudWindow15.Text = "CloudWindow15";
                this.CloudWindow15.Visible = false;
                this.CloudWindow15.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow16
                //
                this.CloudWindow16.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow16.Cloud = null;
                this.CloudWindow16.Location = new System.Drawing.Point(528,
152);
                this.CloudWindow16.Name = "CloudWindow16";
                this.CloudWindow16.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow16.TabIndex = 15;
                this.CloudWindow16.Text = "CloudWindow16";
                this.CloudWindow16.Visible = false;
                this.CloudWindow16.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow17
                //
                this.CloudWindow17.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow17.Cloud = null;
                this.CloudWindow17.Location = new System.Drawing.Point(17,
189);
                this.CloudWindow17.Name = "CloudWindow17";
                this.CloudWindow17.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow17.TabIndex = 16;
                this.CloudWindow17.Text = "CloudWindow17";
                this.CloudWindow17.Visible = false;
                this.CloudWindow17.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow18
                //
                this.CloudWindow18.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow18.Cloud = null;
                this.CloudWindow18.Location = new System.Drawing.Point(157,
189);
                this.CloudWindow18.Name = "CloudWindow18";
                this.CloudWindow18.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow18.TabIndex = 17;
                this.CloudWindow18.Text = "CloudWindow18";
                this.CloudWindow18.Visible = false;
                this.CloudWindow18.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow19
                //
                this.CloudWindow19.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow19.Cloud = null;
                this.CloudWindow19.Location = new System.Drawing.Point(297,
189);
                this.CloudWindow19.Name = "CloudWindow19";
                this.CloudWindow19.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow19.TabIndex = 18;
                this.CloudWindow19.Text = "CloudWindow19";
```

```
                this.CloudWindow19.Visible = false;
                this.CloudWindow19.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow20
                //
                this.CloudWindow20.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow20.Cloud = null;
                this.CloudWindow20.Location = new System.Drawing.Point(17,
261);
                this.CloudWindow20.Name = "CloudWindow20";
                this.CloudWindow20.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow20.TabIndex = 19;
                this.CloudWindow20.Text = "CloudWindow20";
                this.CloudWindow20.Visible = false;
                this.CloudWindow20.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow21
                //
                this.CloudWindow21.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow21.Cloud = null;
                this.CloudWindow21.Location = new System.Drawing.Point(17,
298);
                this.CloudWindow21.Name = "CloudWindow21";
                this.CloudWindow21.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow21.TabIndex = 20;
                this.CloudWindow21.Text = "CloudWindow21";
                this.CloudWindow21.Visible = false;
                this.CloudWindow21.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow22
                //
                this.CloudWindow22.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow22.Cloud = null;
                this.CloudWindow22.Location = new System.Drawing.Point(17,
335);
                this.CloudWindow22.Name = "CloudWindow22";
                this.CloudWindow22.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow22.TabIndex = 21;
                this.CloudWindow22.Text = "CloudWindow22";
                this.CloudWindow22.Visible = false;
                this.CloudWindow22.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow23
                //
                this.CloudWindow23.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow23.Cloud = null;
                this.CloudWindow23.Location = new System.Drawing.Point(17,
335);
                this.CloudWindow23.Name = "CloudWindow23";
                this.CloudWindow23.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow23.TabIndex = 22;
                this.CloudWindow23.Text = "CloudWindow23";
                this.CloudWindow23.Visible = false;
                this.CloudWindow23.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow24
                //
                this.CloudWindow24.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow24.Cloud = null;
```

```
                this.CloudWindow24.Location = new System.Drawing.Point(17,
335);
                this.CloudWindow24.Name = "CloudWindow24";
                this.CloudWindow24.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow24.TabIndex = 23;
                this.CloudWindow24.Text = "CloudWindow24";
                this.CloudWindow24.Visible = false;
                this.CloudWindow24.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // CloudWindow25
                //
                this.CloudWindow25.BackColor =
System.Drawing.SystemColors.ControlDarkDark;
                this.CloudWindow25.Cloud = null;
                this.CloudWindow25.Location = new System.Drawing.Point(17,
335);
                this.CloudWindow25.Name = "CloudWindow25";
                this.CloudWindow25.Size = new System.Drawing.Size(75, 64);
                this.CloudWindow25.TabIndex = 24;
                this.CloudWindow25.Text = "CloudWindow25";
                this.CloudWindow25.Visible = false;
                this.CloudWindow25.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.CloudWindow_MouseDown);
                //
                // Мультиплексер
                //
                this.Controls.AddRange(new System.Windows.Forms.Control[] {

                                this.CloudWindow25,

                                this.CloudWindow24,

                                this.CloudWindow23,

                                this.CloudWindow22,

                                this.CloudWindow21,

                                this.CloudWindow20,

                                this.CloudWindow19,

                                this.CloudWindow18,

                                this.CloudWindow17,

                                this.CloudWindow16,

                                this.CloudWindow15,

                                this.CloudWindow14,

                                this.CloudWindow13,

                                this.CloudWindow12,

                                this.CloudWindow11,

                                this.CloudWindow10,

                                this.CloudWindow9,

                                this.CloudWindow8,

                                this.CloudWindow7,

                                this.CloudWindow6,
```

```
                                this.CloudWindow5,

                                this.CloudWindow4,

                                this.CloudWindow3,

                                this.CloudWindow2,

                                this.CloudWindow1});
                this.Size = new System.Drawing.Size(424, 376);
                this.Resize += new
System.EventHandler(this.Multiplexer_Resize);
                this.ResumeLayout(false);

        }
        #endregion


        // Закриваємо усі хмари
        public void CloseAll()
        {
              for (int i = 0; i < MaxRows; i++)
              {
                    for (int j = 0; j < MaxCols; j++)
                    {
                          camWindows[i, j].Cloud = null;
                    }
              }
        }

        // Беремо зображення з хмари у спеціальну позицію для мультиплексера
        public void SetCloud(int row, int col, Cloud Cloud)
        {
              if ((row >= 0) && (col >= 0) && (row < MaxRows) && (col <
MaxCols))
              {
                    camWindows[row, col].Cloud = Cloud;
              }
        }

        // Встановлюємо розмір  мультиплексера
        public void SetSize(int rows, int cols, int cellWidth, int
cellHeight)
        {
              this.rows = rows;
              this.cols = cols;
              this.cellWidth = cellWidth;
              this.cellHeight = cellHeight;
              UpdateSize();
        }

        // Оновлюємо зображення хмари
        private void UpdateVisiblity()
        {
              if (CloudsVisible)
              {
                    for (int i = 0; i < MaxRows; i++)
                    {
                          for (int j = 0; j < MaxCols; j++)
                          {
                                camWindows[i, j].Visible = ((i < rows) && (j
< cols));
                          }
                    }
              }
        }

        // Оновлюємо розмір та місце хмари
```

```
            private void UpdateSize()
            {
                  int width, height;

                  if (!fitToWindow)
                  {
                        // стандартні ширина та висота
                        width = cellWidth;
                        height = cellHeight;
                  }
                  else
                  {
                        // розраховуємо ширину та висоту хмари для відображення
у вікні
                        width = (ClientRectangle.Width / cols) - 4;
                        height = (ClientRectangle.Height / rows) - 4;
                  }

                  // установлюємо позицію перегляду
                  int startX = (ClientRectangle.Width - cols * (width + 4)) / 2;
                  int startY = (ClientRectangle.Height - rows * (height + 4)) /
2;

                  this.SuspendLayout();

                  for (int i = 0; i < rows; i++)
                  {
                        for (int j = 0; j < cols; j++)
                        {
                              camWindows[i, j].Location = new Point(startX +
(width + 4) * j + 1, startY + (height + 4) * i + 1);
                              camWindows[i, j].Size = new Size(width + 2, height
+ 2);
                        }
                  }

                  this.ResumeLayout(false);
            }

            // Змінюємо розмір
            private void Multiplexer_Resize(object sender, System.EventArgs e)
            {
                  UpdateSize();
            }

            // Миша виключає працю з хмарою
            private void CloudWindow_MouseDown(object sender,
System.Windows.Forms.MouseEventArgs e)
            {
                  lastClicked = (CloudWindow) sender;
            }
      }
}
```

Файл CloudStream.cs – робота з хмарним середовищем у мережі

```
namespace stream
{
	using System;
	using System.Drawing;
	using System.Drawing.Imaging;
	using System.IO;
	using System.Threading;
	using System.Runtime.InteropServices;
	using System.Net;

	using NetCloud;
	using dshow;
	using dshow.Core;

	/// <summary>
	/// CloudStream – поток подачи хмарного середовища у мережі
	/// </summary>
	public class CloudStream : INetCloud
	{
		private string    source;
		private object    userData = null;
		private int       framesReceived;

		private Thread    thread = null;
		private ManualResetEvent stopEvent = null;

		// нова подія у фреймі
		public event CloudEventHandler NewFrame;

		// NetCloud властивості
		public virtual string NetCloud
		{
			get { return source; }
			set { source = value; }
		}
		// Властивості підключення
		public string Login
		{
			get { return null; }
			set { }
		}
		// Властивості паролювання
		public string Password
		{
			get { return null; }
			set {  }
		}
		// FramesReceived властивості
		public int FramesReceived
		{
			get
			{
				int frames = framesReceived;
				framesReceived = 0;
				return frames;
			}
		}
		// BytesReceived властивості
		public int BytesReceived
		{
			get { return 0; }
		}
		// UserData властивості
		public object UserData
		{
```

```
      get { return userData; }
      set { userData = value; }
}
// Отримуємо стан вихідного хмарного середовища у мережі
public bool Running
{
      get
      {
            if (thread != null)
            {
                  if (thread.Join(0) == false)
                        return true;

                  // Якщо стан не заданий, звільнюємо ресурси
                  Free();
            }
            return false;
      }
}


// Конструктор
public CloudStream()
{
}

// Починаємо роботу
public void Start()
{
      if (thread == null)
      {
            framesReceived = 0;

            // Створюємо подію
            stopEvent   = new ManualResetEvent(false);

            // Створюємо й стартуємо нову подію
            thread = new Thread(new ThreadStart(WorkerThread));
            thread.Name = source;
            thread.Start();
      }
}

// Сигнал події до остановки роботи
public void SignalToStop()
{
      // Остановлюємо подію
      if (thread != null)
      {
            // Сигнал остановки
            stopEvent.Set();
      }
}

// Чекаємо остановки події
public void WaitForStop()
{
      if (thread != null)
      {
            // Чекаємо остановки події
            thread.Join();

            Free();
      }
}

// Подія помилки
public void Stop()
{
```

```
            if (this.Running)
            {
                  thread.Abort();
                  // WaitForStop();
            }
      }

      // Визволяємо ресурси
      private void Free()
      {
            thread = null;

            // Випуск події
            stopEvent.Close();
            stopEvent = null;
      }

      // Точка входу події
      public void WorkerThread()
      {
            bool  failed = false;

            // Граббер
            Grabber     grabber = new Grabber(this);

            // Об'єкти
            object      graphObj = null;
            object      sourceObj = null;
            object      grabberObj = null;

            // Інтерфейси
            IGraphBuilder     graph = null;
            IBaseFilter       sourceBase = null;
            IBaseFilter       grabberBase = null;
            ISampleGrabber    sg = null;
            IFileSourceFilter fileSource = null;
            IMediaControl     mc = null;
            IMediaEventEx     mediaEvent = null;

            int   code, param1, param2;

            while ((!failed) && (!stopEvent.WaitOne(0, true)))
            {
                  try
                  {
                        // Встановлюємо тип фільтру графіки
                        Type srvType =
Type.GetTypeFromCLSID(Clsid.FilterGraph);
                        if (srvType == null)
                              throw new ApplicationException("Failed
creating filter graph");

                        // Створюємо фільтр графіки
                        graphObj = Activator.CreateInstance(srvType);
                        graph = (IGraphBuilder) graphObj;

                        // Беремо тип фільтру вікна джерела
                        srvType =
Type.GetTypeFromCLSID(Clsid.WindowsMediaSource);
                        if (srvType == null)
                              throw new ApplicationException("Failed
creating WM source");

                        // Створюємо вікно програвання джерела
                        sourceObj = Activator.CreateInstance(srvType);
                        sourceBase = (IBaseFilter) sourceObj;

                        // Беремо тип простого грабера
```

```
                              srvType =
Type.GetTypeFromCLSID(Clsid.SampleGrabber);
                              if (srvType == null)
                                    throw new ApplicationException("Помилка
створення  простого грабера");

                              // Створення  простого грабера
                              grabberObj = Activator.CreateInstance(srvType);
                              sg = (ISampleGrabber) grabberObj;
                              grabberBase = (IBaseFilter) grabberObj;

                              // Додаємо фільтр графічного джерела
                              graph.AddFilter(sourceBase, "source");
                              graph.AddFilter(grabberBase, "grabber");

                              // Визначаємо тип медіа
                              AMMediaType mt = new AMMediaType();
                              mt.majorType = MediaType.Cloud;
                              mt.subType = MediaSubType.RGB24;
                              sg.SetMediaType(mt);

                              // Редактуємо файл
                              fileSource = (IFileSourceFilter) sourceObj;
                              fileSource.Load(this.source, null);

                              // Підключаємо піни
                              if (graph.Connect(DSTools.GetOutPin(sourceBase,
0), DSTools.GetInPin(grabberBase, 0)) < 0)
                                    throw new ApplicationException("Failed
connecting filters");

                              // Беремо тип медіа
                              if (sg.GetConnectedMediaType(mt) == 0)
                              {
                                    CloudInfoHeader vih = (CloudInfoHeader)
Marshal.PtrToStructure(mt.formatPtr, typeof(CloudInfoHeader));

                                    grabber.Width = vih.BmiHeader.Width;
                                    grabber.Height = vih.BmiHeader.Height;
                                    mt.Dispose();
                              }

                              // рендер
                              graph.Render(DSTools.GetOutPin(grabberBase, 0));

                              //
                              sg.SetBufferSamples(false);
                              sg.SetOneShot(false);
                              sg.SetCallback(grabber, 1);

                              // вікно
                              ICloudWindow win = (ICloudWindow) graphObj;
                              win.put_AutoShow(false);
                              win = null;

                              // Беремо подію інтерфейсу
                              mediaEvent = (IMediaEventEx) graphObj;

                              // Беремо управління медіа
                              mc = (IMediaControl) graphObj;

                              // запускаємо
                              mc.Run();

                              while (!stopEvent.WaitOne(0, true))
                              {
                                    Thread.Sleep(100);

                                    // беремо подію
```

```
                                    if (mediaEvent.GetEvent(out code, out
param1, out param2, 0) == 0)
                                    {
                                          // визначаємо параметри
                                          mediaEvent.FreeEventParams(code,
param1, param2);

                                          //
                                          if (code == (int) EventCode.Complete)
                                          {
                                                break;
                                          }
                                    }
                              }

                              mc.StopWhenReady();
                        }
                              // Визначаємо виключення
                        catch (Exception e)
                        {
                              System.Diagnostics.Debug.WriteLine("----: " +
e.Message);
                              failed = true;
                        }
                              // Фіналізуємо блок
                        finally
                        {
                              // визначаємо усі об'єкти
                              mediaEvent = null;
                              mc = null;
                              fileSource = null;
                              graph = null;
                              sourceBase = null;
                              grabberBase = null;
                              sg = null;

                              if (graphObj != null)
                              {
                                    Marshal.ReleaseComObject(graphObj);
                                    graphObj = null;
                              }
                              if (sourceObj != null)
                              {
                                    Marshal.ReleaseComObject(sourceObj);
                                    sourceObj = null;
                              }
                              if (grabberObj != null)
                              {
                                    Marshal.ReleaseComObject(grabberObj);
                                    grabberObj = null;
                              }
                        }
                  }
            }

            // новий фрейм для обробки
            protected void OnNewFrame(Bitmap image)
            {
                  framesReceived++;
                  if (NewFrame != null)
                        NewFrame(this, new CloudEventArgs(image));
            }

            // Граббер
            private class Grabber : ISampleGrabberCB
            {
                  private CloudStream parent;
                  private int width, height;
```

```
                // Width властивості
                public int Width
                {
                        get { return width; }
                        set { width = value; }
                }
                // Height властивості
                public int Height
                {
                        get { return height; }
                        set { height = value; }
                }

                // Конструктор
                public Grabber(CloudStream parent)
                {
                        this.parent = parent;
                }

                //
                public int SampleCB(double SampleTime, IntPtr pSample)
                {
                        return 0;
                }

                // Повертаємо метод, який вказує на буфер взірця
                public int BufferCB(double SampleTime, IntPtr pBuffer, int
BufferLen)
                {
                        // створюємо нову картинку
                        System.Drawing.Bitmap img = new Bitmap(width, height,
PixelFormat.Format24bppRgb);

                        // блокуємо дані бітової площини
                        BitmapData  bmData = img.LockBits(
                                new Rectangle(0, 0, width, height),
                                ImageLockMode.ReadWrite,
                                PixelFormat.Format24bppRgb);

                        // копіюємо дані зображення
                        int srcStride = bmData.Stride;
                        int dstStride = bmData.Stride;

                        int dst = bmData.Scan0.ToInt32() + dstStride * (height -
1);
                        int src = pBuffer.ToInt32();

                        for (int y = 0; y < height; y++)
                        {
                                Win32.memcpy(dst, src, srcStride);
                                dst -= dstStride;
                                src += srcStride;
                        }

                        // розблокуємо дані бітової площини
                        img.UnlockBits(bmData);

                        // Увідомляємо батьків
                        parent.OnNewFrame(img);

                        // Будуємо картинку
                        img.Dispose();

                        return 0;
                }
        }
    }
}
```

Файл CloudStreamSetupPage.cs – робота з хмарним середовищем у мережі (інтерфейс)

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;
using NetCloud;

namespace stream
{
	/// <summary>
	/// Основні дескриптори для CloudStreamSetupPage.
	/// </summary>
	public class CloudStreamSetupPage : System.Windows.Forms.UserControl,
INetCloudPage
	{
		private bool completed = false;
		private System.Windows.Forms.TextBox urlBox;
		private System.Windows.Forms.Label label1;
		/// <summary>
		/// Опис змінних розробника.
		/// </summary>
		private System.ComponentModel.Container components = null;

		// стан змінюваної події
		public event EventHandler StateChanged;

		// Конструктор
		public CloudStreamSetupPage()
		{
			// Цей виклик використовується у Windows.Forms Form Designer.
			InitializeComponent();

		}

		/// <summary>
		/// Очищуємо усі ресурси використовувані користувачем.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Component Designer generated code
		/// <summary>
		/// Необхідний метод для підтримки розробника - не модифікується
		/// контент цього методу з редактором коду.
		/// </summary>
		private void InitializeComponent()
		{
			this.urlBox = new System.Windows.Forms.TextBox();
			this.label1 = new System.Windows.Forms.Label();
			this.SuspendLayout();
			//
			// urlBox
			//
			this.urlBox.Anchor = ((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Left)
```

```
                        | System.Windows.Forms.AnchorStyles.Right);
                this.urlBox.Location = new System.Drawing.Point(50, 10);
                this.urlBox.Name = "urlBox";
                this.urlBox.Size = new System.Drawing.Size(240, 20);
                this.urlBox.TabIndex = 1;
                this.urlBox.Text = "";
                this.urlBox.TextChanged += new
System.EventHandler(this.urlBox_TextChanged);
                //
                // label1
                //
                this.label1.Location = new System.Drawing.Point(10, 13);
                this.label1.Name = "label1";
                this.label1.Size = new System.Drawing.Size(30, 14);
                this.label1.TabIndex = 0;
                this.label1.Text = "&URL:";
                //
                // CloudStreamSetupPage
                //
                this.Controls.AddRange(new System.Windows.Forms.Control[] {

                                this.urlBox,

                                this.label1});
                this.Name = "CloudStreamSetupPage";
                this.Size = new System.Drawing.Size(300, 150);
                this.ResumeLayout(false);

        }
        #endregion

        // Completed властивості
        public bool Completed
        {
                get { return completed; }
        }

        // Показуємо сторінку
        public void Display()
        {
                urlBox.Focus();
                urlBox.SelectionStart = urlBox.TextLength;
        }

        // Додаємо сторінку
        public bool Apply()
        {
                return true;
        }

        // Конфігуруємо об’єкт зображення
        public object GetConfiguration()
        {
                StreamConfiguration config = new StreamConfiguration();

                config.source = urlBox.Text;

                return (object) config;
        }

        // Встановлюємо конфігурацію
        public void SetConfiguration(object config)
        {
                StreamConfiguration cfg = (StreamConfiguration) config;

                if (cfg != null)
                {
                        urlBox.Text = cfg.source;
                }
```

```
            }

            // Змінюємо URL
            private void urlBox_TextChanged(object sender, System.EventArgs e)
            {
                  completed = (urlBox.TextLength != 0);

                  if (StateChanged != null)
                        StateChanged(this, new EventArgs());
            }
      }
}
```

Файл BinomialCodeSource.cs – алгоритм рівноважного кодування на основі біноміальних чисел

```
namespace BinomialCode
{
	using System;
	using System.Drawing;
	using System.IO;
	using System.Threading;
	using System.Net;

	using NetCloud;

	/// <summary>
	/// BinomialCodeSource - BinomialCode скачувач
	/// </summary>
	public class BinomialCodeSource : INetCloud
	{
		private string    source;
		private string    login = null;
		private string    password = null;
		private object    userData = null;
		private int       framesReceived;
		private int       bytesReceived;
		private bool      useSeparateConnectionGroup = false;
		private bool      preventCaching = false;
		private int       frameInterval = 0;          // інтервал подання
фреймів у мілісекундах

		private const int bufSize = 512 * 1024;   // розмір буферу
		private const int readSize = 1024;        // розмір блоку для читання

		private Thread    thread = null;
		private ManualResetEvent stopEvent = null;

		// нова подія у фреймі
		public event CloudEventHandler NewFrame;

		// SeparateConnectioGroup властивості
		// indicates to open WebRequest in separate connection group
		public bool SeparateConnectionGroup
		{
			get { return useSeparateConnectionGroup; }
			set { useSeparateConnectionGroup = value; }
		}
		// PreventCaching властивості
		// Якщо властивості є правильними, то керуємо параметриу URL. Цей
клієнт повинен бути встановлений на проксі-сервері.
		public bool PreventCaching
		{
			get { return preventCaching; }
			set { preventCaching = value; }
		}
		// FrameInterval властивості - інтервал між фреймами
		Якщо властивості встановлені в 100, тоді джерело формує 10 фреймів
		// в секунду
		public int FrameInterval
		{
			get { return frameInterval; }
			set { frameInterval = value; }
		}
		// NetCloud властивості
		public virtual string NetCloud
		{
			get { return source; }
			set { source = value; }
		}
```

```
// Властивості підключення
public string Login
{
      get { return login; }
      set { login = value; }
}
// Властивості паролювання
public string Password
{
      get { return password; }
      set { password = value; }
}
// FramesReceived властивості
public int FramesReceived
{
      get
      {
            int frames = framesReceived;
            framesReceived = 0;
            return frames;
      }
}
// BytesReceived властивості
public int BytesReceived
{
      get
      {
            int bytes = bytesReceived;
            bytesReceived = 0;
            return bytes;
      }
}
// UserData властивості
public object UserData
{
      get { return userData; }
      set { userData = value; }
}
// Отримуємо стан вихідного хмарного середовища у мережі
public bool Running
{
      get
      {
            if (thread != null)
            {
                  if (thread.Join(0) == false)
                        return true;

                  // Якщо стан не заданий, звільнюємо ресурси
                  Free();
            }
            return false;
      }
}

// Конструктор
public BinomialCodeSource()
{
}

// Починаємо роботу
public void Start()
{
      if (thread == null)
      {
            framesReceived = 0;
            bytesReceived = 0;

            // Створюємо подію
```

```
                        stopEvent   = new ManualResetEvent(false);

                        // Створюємо й стартуємо нову подію
                        thread = new Thread(new ThreadStart(WorkerThread));
                        thread.Name = source;
                        thread.Start();
                  }
            }

            // Сигнал події до остановки роботи
            public void SignalToStop()
            {
                  // Остановлюємо подію
                  if (thread != null)
                  {
                        // Сигнал остановки
                        stopEvent.Set();
                  }
            }

            // Чекаємо остановки події
            public void WaitForStop()
            {
                  if (thread != null)
                  {
                        // Чекаємо остановки події
                        thread.Join();

                        Free();
                  }
            }

            // Подія помилки
            public void Stop()
            {
                  if (this.Running)
                  {
                        thread.Abort();
                        WaitForStop();
                  }
            }

            // Визволяємо ресурси
            private void Free()
            {
                  thread = null;

                  // Випуск події
                  stopEvent.Close();
                  stopEvent = null;
            }

            // Точка входу події
            public void WorkerThread()
            {
                  byte[]                  buffer = new byte[bufSize];   // буфер
читання потоку
                  HttpWebRequest    req = null;
                  WebResponse       resp = null;
                  Stream                  stream = null;
                  Random                  rnd = new Random((int)
DateTime.Now.Ticks);
                  DateTime          start;
                  TimeSpan          span;

                  while (true)
                  {
                        int   read, total = 0;
```

```
                        try
                        {
                              start = DateTime.Now;

                              // створюємо запрос
                              if (!preventCaching)
                              {
                                    req = (HttpWebRequest)
WebRequest.Create(source);
                              }
                              else
                              {
                                    req = (HttpWebRequest)
WebRequest.Create(source + ((source.IndexOf('?') == -1) ? '?' : '&') + "fake=" +
rnd.Next().ToString());
                              }
                              // встановлюємо логін та пароль
                              if ((login != null) && (password != null) &&
(login != ""))
                                    req.Credentials = new
NetworkCredential(login, password);
                              // встановлюємо найменування группи підключення
                              if (useSeparateConnectionGroup)
                                    req.ConnectionGroupName =
GetHashCode().ToString();
                              // отримуємо відповідь
                              resp = req.GetResponse();

                              // отримуємо відповідь потоку
                              stream = resp.GetResponseStream();

                              // цикл
                              while (!stopEvent.WaitOne(0, true))
                              {
                                    // перевіряємо загальне читання
                                    if (total > bufSize - readSize)
                                    {
                                          total = 0;
                                    }

                                    // Читаємо наступний блок у потоці
                                    if ((read = stream.Read(buffer, total,
readSize)) == 0)
                                          break;
                                    total += read;
                                    // Додаємо лічильник зчитаних байт
                                    bytesReceived += read;
                              }
                              if (!stopEvent.WaitOne(0, true))
                              {
                                    // додаємо лічильник фреймів
                                    framesReceived++;
                                    // остановка читання зображення
                                    if (NewFrame != null)
                                    {
                                          Bitmap      bmp = (Bitmap)
Bitmap.FromStream(new MemoryStream(buffer, 0, total));
                                          // Клієнт увідомлення
                                          NewFrame(this, new
CloudEventArgs(bmp));
                                          // Будуємо картинку
                                          bmp.Dispose();
                                          bmp = null;
                                    }
                              }
                              // Чекаємо в циклі ?
                              if (frameInterval > 0)
                              {
                                    // діапазон часу
```

```
                                    span = DateTime.Now.Subtract(start);
                                    // засипання у мілісекундах
                                    int msec = frameInterval - (int)
span.TotalMilliseconds;
                                    while ((msec > 0) && (stopEvent.WaitOne(0,
true) == false))
                                    {
                                          // засипання ...
                                          Thread.Sleep((msec < 100) ? msec :
100);
                                          msec -= 100;
                                    }
                              }
                        }
                        catch (WebException ex)
                        {
                              System.Diagnostics.Debug.WriteLine("=============:
" + ex.Message);
                              // чекаємо у циклі наступну спробу
                              Thread.Sleep(250);
                        }
                        catch (Exception ex)
                        {
                              System.Diagnostics.Debug.WriteLine("=============:
" + ex.Message);
                        }
                        finally
                        {
                              // помилка запиту
                              if (req != null)
                              {
                                    req.Abort();
                                    req = null;
                              }
                              // закриваємо потоку
                              if (stream != null)
                              {
                                    stream.Close();
                                    stream = null;
                              }
                              // close response
                              if (resp != null)
                              {
                                    resp.Close();
                                    resp = null;
                              }
                        }
                        // need to stop ?
                        if (stopEvent.WaitOne(0, true))
                              break;
                  }
            }
      }
}
```

Файл BinomialCodeSourcePage.cs – алгоритм рівноважного кодування на основі біноміальних чисел (інтерфейс)

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;
using NetCloud;

namespace BinomialCode
{
	/// <summary>
	/// Основні дескриптори для BinomialCodeSourcePage.
	/// </summary>
	public class BinomialCodeSourcePage : System.Windows.Forms.UserControl,
INetCloudPage
	{
		private static int[] frameIntervals = new int[] {0, 100, 142, 200,
333, 1000,
                                    5000, 10000, 15000, 20000, 30000, 60000};
		private bool completed = false;
		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.TextBox urlBox;
		private System.Windows.Forms.Label label2;
		private System.Windows.Forms.TextBox loginBox;
		private System.Windows.Forms.Label label3;
		private System.Windows.Forms.TextBox passwordBox;
		private System.Windows.Forms.Label label4;
		private System.Windows.Forms.ComboBox rateCombo;
		/// <summary>
		/// Опис змінних розробника.
		/// </summary>
		private System.ComponentModel.Container components = null;

		// стан змінюваної події
		public event EventHandler StateChanged;

		// Конструктор
		public BinomialCodeSourcePage()
		{
			// Цей виклик використовується у Windows.Forms Form Designer.
			InitializeComponent();

			//
			rateCombo.SelectedIndex = 0;
		}

		/// <summary>
		/// Очищуємо усі ресурси використовувані користувачем.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Component Designer generated code
		/// <summary>
		/// Необхідний метод для підтримки розробника - не модифікується
```

```
            /// контент цього методу з редактором коду.
            /// </summary>
            private void InitializeComponent()
            {
                  this.label1 = new System.Windows.Forms.Label();
                  this.urlBox = new System.Windows.Forms.TextBox();
                  this.label2 = new System.Windows.Forms.Label();
                  this.loginBox = new System.Windows.Forms.TextBox();
                  this.label3 = new System.Windows.Forms.Label();
                  this.passwordBox = new System.Windows.Forms.TextBox();
                  this.label4 = new System.Windows.Forms.Label();
                  this.rateCombo = new System.Windows.Forms.ComboBox();
                  this.SuspendLayout();
                  //
                  // label1
                  //
                  this.label1.Location = new System.Drawing.Point(10, 13);
                  this.label1.Name = "label1";
                  this.label1.Size = new System.Drawing.Size(41, 14);
                  this.label1.TabIndex = 0;
                  this.label1.Text = "&URL:";
                  //
                  // urlBox
                  //
                  this.urlBox.Anchor = ((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right);
                  this.urlBox.Location = new System.Drawing.Point(70, 10);
                  this.urlBox.Name = "urlBox";
                  this.urlBox.Size = new System.Drawing.Size(220, 20);
                  this.urlBox.TabIndex = 1;
                  this.urlBox.Text = "";
                  this.urlBox.TextChanged += new
System.EventHandler(this.urlBox_TextChanged);
                  //
                  // label2
                  //
                  this.label2.Location = new System.Drawing.Point(10, 43);
                  this.label2.Name = "label2";
                  this.label2.Size = new System.Drawing.Size(35, 14);
                  this.label2.TabIndex = 2;
                  this.label2.Text = "&Login:";
                  //
                  // loginBox
                  //
                  this.loginBox.Anchor = ((System.Windows.Forms.AnchorStyles.Top
| System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right);
                  this.loginBox.Location = new System.Drawing.Point(70, 40);
                  this.loginBox.Name = "loginBox";
                  this.loginBox.Size = new System.Drawing.Size(220, 20);
                  this.loginBox.TabIndex = 3;
                  this.loginBox.Text = "";
                  //
                  // label3
                  //
                  this.label3.Location = new System.Drawing.Point(10, 73);
                  this.label3.Name = "label3";
                  this.label3.Size = new System.Drawing.Size(60, 14);
                  this.label3.TabIndex = 4;
                  this.label3.Text = "&Password:";
                  //
                  // passwordBox
                  //
                  this.passwordBox.Anchor =
((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right);
                  this.passwordBox.Location = new System.Drawing.Point(70, 70);
```

```
                this.passwordBox.Name = "passwordBox";
                this.passwordBox.Size = new System.Drawing.Size(220, 20);
                this.passwordBox.TabIndex = 5;
                this.passwordBox.Text = "";
                //
                // label4
                //
                this.label4.Location = new System.Drawing.Point(10, 103);
                this.label4.Name = "label4";
                this.label4.Size = new System.Drawing.Size(63, 14);
                this.label4.TabIndex = 6;
                this.label4.Text = "&Frame rate:";
                //
                // rateCombo
                //
                this.rateCombo.Anchor =
((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right);
                this.rateCombo.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList;
                this.rateCombo.Items.AddRange(new object[] {

        "Uncontrolled",

        "10 фреймів у секунду",

        "7 фреймів у секунду",

        "5 фреймів у секунду",

        "3 фреймів у секунду",

        "1 фреймів у секунду",

        "12 фреймів у хвилину",

        "6 фреймів у хвилину",

        "4 фреймів у хвилину",

        "3 фреймів у хвилину",

        "2 фреймів у хвилину",

        "1 фреймів у хвилину"});
                this.rateCombo.Location = new System.Drawing.Point(70, 100);
                this.rateCombo.Name = "rateCombo";
                this.rateCombo.Size = new System.Drawing.Size(220, 21);
                this.rateCombo.TabIndex = 7;
                //
                // BinomialCodeSourcePage
                //
                this.Controls.AddRange(new System.Windows.Forms.Control[] {

                                this.rateCombo,

                                this.label4,

                                this.passwordBox,

                                this.label3,

                                this.loginBox,

                                this.label2,

                                this.urlBox,
```

```
                              this.label1});
                  this.Name = "BinomialCodeSourcePage";
                  this.Size = new System.Drawing.Size(300, 150);
                  this.ResumeLayout(false);

            }
            #endregion

            // Completed властивості
            public bool Completed
            {
                  get { return completed; }
            }

            // Показуємо сторінку
            public void Display()
            {
                  urlBox.Focus();
                  urlBox.SelectionStart = urlBox.TextLength;
            }

            // Додаємо сторінку
            public bool Apply()
            {
                  return true;
            }

            // Конфігуруємо об’єкт зображення
            public object GetConfiguration()
            {
                  BinomialCodeConfiguration    config = new
BinomialCodeConfiguration();

                  config.source     = urlBox.Text;
                  config.login      = loginBox.Text;
                  config.password   = passwordBox.Text;
                  config.frameInterval =
frameIntervals[rateCombo.SelectedIndex];

                  return (object) config;
            }

            // Встановлюємо конфігурацію
            public void SetConfiguration(object config)
            {
                  BinomialCodeConfiguration    cfg = (BinomialCodeConfiguration)
config;

                  if (cfg != null)
                  {
                        urlBox.Text = cfg.source;
                        loginBox.Text = cfg.login;
                        passwordBox.Text = cfg.password;
                        rateCombo.SelectedIndex = Array.IndexOf(frameIntervals,
cfg.frameInterval);
                  }
            }

            // Змінюємо URL
            private void urlBox_TextChanged(object sender, System.EventArgs e)
            {
                  completed = (urlBox.TextLength != 0);

                  if (StateChanged != null)
                        StateChanged(this, new EventArgs());
            }
      }
}
```

Файл About.cs - довідка

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace CloudAbout {
      public class About : System.Windows.Forms.Form {

            #region system stuff
            private System.Windows.Forms.PictureBox pictureBox1;
            private System.Windows.Forms.RichTextBox richTextBox1;
            private System.Windows.Forms.Button button1;
            private System.ComponentModel.Container components = null;

            public About() {
                  InitializeComponent();
            }

            protected override void Dispose( bool disposing ) {
                  if( disposing ) {
                        if(components != null) {
                              components.Dispose();
                        }
                  }
                  base.Dispose( disposing );
            }
            #endregion

            #region Windows Form Designer generated code
            /// <summary>
            /// Необхідний метод для підтримки розробника - не модифікується
            /// контент цього методу з редактором коду.
            /// </summary>
            private void InitializeComponent() {
                  System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(About));
                  this.pictureBox1 = new System.Windows.Forms.PictureBox();
                  this.richTextBox1 = new System.Windows.Forms.RichTextBox();
                  this.button1 = new System.Windows.Forms.Button();
                  this.SuspendLayout();
                  //
                  // pictureBox1
                  //
                  this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
                  this.pictureBox1.Location = new System.Drawing.Point(8, 8);
                  this.pictureBox1.Name = "pictureBox1";
                  this.pictureBox1.Size = new System.Drawing.Size(152, 192);
                  this.pictureBox1.TabIndex = 0;
                  this.pictureBox1.TabStop = false;
                  //
                  // richTextBox1
                  //
                  this.richTextBox1.BackColor = System.Drawing.Color.Black;
                  this.richTextBox1.BorderStyle =
System.Windows.Forms.BorderStyle.None;
                  this.richTextBox1.ForeColor = System.Drawing.Color.White;
                  this.richTextBox1.Location = new System.Drawing.Point(176,
16);
                  this.richTextBox1.Name = "richTextBox1";
                  this.richTextBox1.Size = new System.Drawing.Size(304, 184);
                  this.richTextBox1.TabIndex = 1;
                  this.richTextBox1.Text = @"БАКАЛАВРСЬКСЬКА РОБОТА
```

на тему:

Програмне забезпечення системи адаптивного рівноважного кодування на основі біноміальних чисел для хмарних технологій

Керівник: Усік П.С.

Розробив: студент Трущ Микита Олександрович
гр. КІ-21-1

м. Кропивницький 2025";

```
                    //
                    // button1
                    //
                    this.button1.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
                    this.button1.ForeColor = System.Drawing.Color.White;
                    this.button1.Location = new System.Drawing.Point(16, 168);
                    this.button1.Name = "button1";
                    this.button1.Size = new System.Drawing.Size(136, 24);
                    this.button1.TabIndex = 2;
                    this.button1.Text = "&Close";
                    this.button1.Click += new
System.EventHandler(this.button1_Click);
                    //
                    // About
                    //
                    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
                    this.BackColor = System.Drawing.Color.Black;
                    this.ClientSize = new System.Drawing.Size(480, 208);
                    this.Controls.Add(this.button1);
                    this.Controls.Add(this.richTextBox1);
                    this.Controls.Add(this.pictureBox1);
                    this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.None;
                    this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
                    this.Name = "About";
                    this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
                    this.Text = "About";
                    this.ResumeLayout(false);

            }
            #endregion

            #region events

            private void button1_Click(object sender, System.EventArgs e) {
                  this.Close();
            }

            #endregion
      }
}
```