Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка програмного застосунку для ущільнення зображень з використанням рекурсивного методу»

Виконав: студент 4 курсу
групи 1ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Слободян Д.І.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Майданюк В.П.
(прізвище та ініціали)
Рецензент: д.ф., ст. викл. Обертюх М. Р.
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О.Н.
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н._____
«21» березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Слободяну Денису Івановичу

1. Тема роботи – «Розробка програмного застосунку для ущільнення зображень з використанням рекурсивного методу»

Керівник роботи: Майданюк Володимир Павлович, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.

2. Строк подання студентом роботи 2 червня 2025 р.

3. Вихідні дані до роботи: тип перетворення – DWT; операційна система – Windows; середовище розробки – MS Visual Studio; мова програмування – C#, коефіцієнт ущільнення - 2.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану розвитку програмних застосунків для ущільнення зображень та обгрунтування задач розробки; розробка застосунку; тестування застосунку; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження, наукова новизна, практична цінність одержаних результатів; порівняльний аналіз аналогів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Майданюк В.П., к.т.н., доцент кафедри ПЗ	24.03.25	02.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку програмних застосунків для ущільнення зображень	25.03.25 – 31.03.25	вип
2	Розробка методів, моделі та алгоритму роботи програмного забезпечення	03.04.25 – 14.04.25	вип
3	Варіантний аналіз та вибір мови програмування розробки	17.04.25 – 21.04.25	вип
4	Розробка програмного забезпечення	24.04.25 – 12.05.25	вип
5	Тестування програмного забезпечення	15.05.25 – 19.05.25	вип
6	Оформлення матеріалів до захисту БКР	22.05.25 – 30.05.25	вип

Студент _____ Слободян Д.І.
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ Майданюк В.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.4

Слободян Д. І. Розробка програмного застосунку для ущільнення зображень з використанням рекурсивного методу : бакалаврська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця : ВНТУ, 2025. 58 с.

На укр. мові. Бібліогр. : 29 назв ; рис. : 28 ; табл. 3.

У бакалаврській кваліфікаційній роботі проведено аналіз сучасних методів ущільнення зображень та особливостей їх застосування в умовах збереження якості. Розглянуто переваги використання рекурсивних підходів, що базуються на ієрархічному поділі зображення з подальшим кодуванням його компонентів.

Розроблено програмне забезпечення для ущільнення зображень з використанням рекурсивного методу, реалізоване в середовищі Microsoft Visual Studio мовою програмування C#. Програмний застосунок має графічний інтерфейс користувача та підтримує налаштування параметрів ущільнення. Це дає змогу адаптувати процес обробки під конкретні вимоги до якості та розміру зображення.

Результати дослідження свідчать про доцільність використання рекурсивного методу як ефективного підходу до ущільнення, що дозволяє зменшити обчислювальну складність і підвищити адаптивність до структури зображення. Розроблене програмне забезпечення може бути використане як у наукових дослідженнях, так і в освітньому процесі при вивченні методів цифрової обробки зображень.

Ключові слова: Windows, Visual Studio, C#, ущільнення зображень, рекурсивний метод, графічний інтерфейс.

ABSTRACT

UDC 004.4

Slobodyan D. I. Development of an Image Compression Application Using a Recursive Method. Vinnytsia :VNTU, 2025. 58 p.

In Ukrainian. Bibliography: 26 titles; figures: 28; tables: 3.

This bachelor's qualification thesis presents an analysis of modern image compression methods and the specifics of their application while preserving visual quality. The advantages of using recursive approaches based on hierarchical image division and subsequent component-wise encoding are explored.

An image compression software application based on a recursive method has been developed using the Microsoft Visual Studio environment and the C# programming language. The application features a graphical user interface and supports the adjustment of compression parameters, enabling the adaptation of the processing to specific requirements regarding image quality and size.

The research results demonstrate the feasibility of using the recursive method as an effective approach to image compression, which allows reducing computational complexity and increasing adaptability to the structural characteristics of the image. The developed software can be used in both scientific research and educational contexts when studying digital image processing and encoding techniques.

Keywords: Windows, Visual Studio, C#, image compression, recursive method, graphical user interface.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному світі стрімке зростання обсягів графічної інформації потребує ефективних методів її зберігання, передавання та обробки. З огляду на це, питання ущільнення зображень набуває особливої актуальності.

Алгоритми ущільнення без втрат інформації (кодування Хаффмана, LZW-кодування, арифметичне кодування) не можуть забезпечити високий коефіцієнт ущільнення зображень, оскільки зображення – це джерело повідомлень з рівномірним розподілом [1]. Єдиним способом ефективного ущільнення зображень є застосування алгоритмів з втратами інформації.

Серед алгоритмів з втратою інформації фактичним стандартом є алгоритм JPEG, який підтримується багаточисленними програмними та апаратними системами. Проте цей алгоритм має ряд недоліків: складні обчислення при виконанні ущільнення і відновлення, накопичення дефектів при виконанні повторного ущільнення. Крім того, JPEG не завжди забезпечує оптимальне співвідношення між якістю та ступенем стиснення. Через ці недоліки зображення, ущільнені відповідно до стандарту JPEG, не можна використовувати у певних випадках. Саме тому актуальним є розвиток та дослідження нових алгоритмів ущільнення зображень.

Одним з перспективних алгоритмів ущільнення зображень є алгоритм з використанням Wavelet-перетворень, який може забезпечити коефіцієнт ущільнення не нижчий ніж JPEG, проте має перед ним ряд переваг. До цих переваг можна віднести високу швидкість виконання ущільнення та відновлення та просту апаратну реалізацію. Велика швидкодія алгоритму досягається завдяки простоті математичних операцій. При кодуванні алгоритм на основі Wavelet-перетворень використовує лише додавання та ділення на два. Додавання та ділення – це найпростіші операції з точки зору комп’ютерної арифметики. Ще однією перевагою у порівнянні з стандартом JPEG є можливість повторного ущільнення відновленого зображення без накопичення дефектів. Завдяки цьому

зображення, ущільнене з використанням Wavelet-перетворень, можна застосовувати там, де потрібно виконувати редагування зображення дуже багато разів. Саме тому було обрано тему «Розробка програмного застосунку для ущільнення зображень з використанням рекурсивного методу».

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою бакалаврської кваліфікаційної роботи є підвищення коефіцієнту ущільнення зображень з використанням рекурсивного перетворення.

Для досягнення поставленої мети були визначені наступні завдання:

– провести аналіз існуючих методів ущільнення зображень;
– обґрунтувати вибір рекурсивного підходу до ущільнення;
– розробити алгоритм рекурсивного ущільнення на основі розбиття зображення;
– реалізувати програмний застосунок з графічним інтерфейсом користувача;
– провести тестування ефективності розробленого рішення.

Об'єкт дослідження – процес ущільнення растрових зображень.

Предмет дослідження – – методи та програмні засоби ущільнення зображень на основі рекурсивного перетворення.

Методи дослідження. У процесі досліджень використовувались: теорія алгоритмів, теорія цифрової обробки зображень для розробки моделей та методів ущільнення зображень на основі рекурсивного перетворення; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод метод ущільнення зображень на основі Wavelet-перетворення Хаара, у якому, на відміну від існуючих, кількість компонент розкладу зображення збільшено до 10 за рахунок повторного розкладу низькочастотної компоненти зображення, що підвищує коефіцієнт ущільнення в

1,3 рази.

2. Подальшого розвитку отримав метод квантування компонент зображення у якому, на відміну від існуючих, високочастотні компоненти квантуються з використанням логарифмічних шкал, що дозволяє зменшити кількість біт на піксель у 2 рази.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено мовою програмування C# програмні засоби для дослідження ущільнення зображень з використанням рекурсивного методу.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто. У друкованій праці, опублікованій у співавторстві, автору належать такі результати: схема виконання Wavelet-перетворення [2].

Апробація результатів роботи. Основні положення бакалаврської кваліфікаційної роботи доповідалися та обговорювалися на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)».

Публікації. Основні результати досліджень опубліковано в одній статті у матеріалах конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)».

Аналіз. У пояснювальній записці до бакалаврської кваліфікаційної роботи було розглянуто 4 розділи та було використано 29 літературних джерел.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ПРОГРАМНИХ ЗАСТОСУНКІВ ДЛЯ УЩІЛЬНЕННЯ ЗОБРАЖЕНЬ ТА ОБҐРУНТУВАННЯ ЗАДАЧ РОЗРОБКИ

1.1 Основні вимоги до алгоритмів ущільнення зображень

Зображення – це своєрідний тип даних, що характеризується трьома особливостями :

1. Зображення (як і відео) займає набагато більше місця в пам'яті, ніж текст. Для прикладу, не дуже якісна ілюстрація на обкладинці книги розміром 500x800 пікселів, займає 1.2 Мб - стільки ж, скільки художня книга з 400 сторінок (60 знаків у рядку, 42 рядки на сторінці). Як приклад можна розглянути також, скільки тисяч сторінок тексту ми зможемо помістити на CD-диску, і як мало там поміститься якісних ущільнених зображень. Ця особливість зображень визначає актуальність алгоритмів ущільнення графічних зображень.

2. Другою особливістю зображень є те, що людський зір при аналізі зображення оперує контурами, загальним переходом кольорів і порівняно невідчутно до малих змін в зображенні. Таким чином, можна створити ефективні алгоритми ущільнення зображень, в яких відновлене після ущільнення зображення буде збігатися з оригіналом при розгляді його людиною, хоча у відновленому зображенні будуть незначні дефекти. Дана особливість людського зору дозволила створити спеціальні алгоритми стиснення, орієнтовані тільки на зображення. Ці алгоритми мають дуже високі коефіцієнти ущільнення.

3. Зображення на відміну від тексту, володіють надлишковістю в 2-х вимірах. Тобто, як правило, сусідні точки, як по горизонталі, так і по вертикалі, в зображенні близькі за кольором. Крім того, можна використовувати подібність між колірними площинами R, G і B в алгоритмах ущільнення, що дає можливість створити ще більш ефективні алгоритми. Таким чином, при створенні алгоритму ущільнення графічних зображень використовуються особливості структури зберігання [3].

Алгоритми ущільнення дають хороший результат при врахуванні усіх особливостей графічних зображень.

До алгоритмів ущільнення висуваються такі вимоги:

1. високий коефіцієнт ущільнення,
2. низькі втрати при виконанні ущільнення,
3. висока швидкість ущільнення та відновлення ущільненого зображення,
4. масштабування зображень,
5. можливість показати зменшену копію зображення при перегляді лише початку файлу,
6. стійкість до помилок,
7. врахування специфічних особливостей зображень,
8. можливість редагування,
9. невисока вартість апаратної та програмної реалізації.

Високий коефіцієнт ущільнення є головною вимогою до алгоритмів ущільнення зображень. Але не у всіх випадках високий коефіцієнт ущільнення є головним критерієм. Заради високого коефіцієнта ущільнення доводиться жертвувати якістю відновленого зображення, швидкістю виконання ущільнення та відновлення та великими затратами на програмну та апаратну реалізацію . Коефіцієнт ущільнення знаходиться за формулою 1.1 і показує, у скільки раз ущільнене зображення менше неущільненого [4].

$$K_y = \frac{V_1}{V_2} \tag{1.1}$$

де V_1 – розмір початкового зображення, V_2 – розмір ущільненого зображення.

Низькі втрати при виконанні ущільнення зображення позитивно відображаються на якості відновленого зображення. Ця вимога є основною при ущільненні таких зображень, на яких велика кількість важливої інформації зберігається у дрібних деталях, спотворення яких не зможе передати інформацію, яка містилася на початковому зображенні. Прикладом такого зображення може

бути, наприклад, медичний знімок. Одна з проблем машинної графіки полягає в тому, що на даний час не знайдений адекватний критерій оцінки втрат якості зображення. Якість втрачається при оцифровуванні, при перекладі в обмежену палітру кольорів або в інший колірний простір, а також при стисненні зображень з втратами.

Нехай є два зображення: f – початкове зображення і $f^{|}$ – відновлене після ущільнення зображення [5]. Одними з критеріїв оцінки втрати якості є середнє квадратичне відхилення значень пікселів відновленого після ущільнення зображення від початкового зображення, яке знаходиться за формулою 1.2 та дисперсія, яка знаходиться за формулою 1.3.

$$\sigma = \frac{\sum_{x,y}^{M,N}(f(x,y) - f^{|}(x,\ y))^2}{M*N} \tag{1.2}$$

де M – ширина зображення в пікселях, N – висота зображення в піселях, x та y координати пікселя на зображенні.

$$d = \sqrt{\frac{\sum_{x,y}^{M,N}(f(x,y) - f^{|}(x,\ y))^2}{M*N}} = \sqrt{\sigma} \tag{1.3}$$

Використовуючи для оцінки середнє квадратичне відхилення або дисперсію, зображення буде сильно зіпсовано при зміні яскравості всього на 5% [6]. У той же час зображення з різними шумами, різкими змінами кольору окремих точок будуть визнаватися як майже не спотворені, хоча для людського зору буде чітко видно дефекти зображення.

Іншим критерієм є максимальне відхилення від оригіналу, яке визначається за формулою 1.4. Даний критерій дуже чутливий до сильного спотворення окремих пікселів. Зображення може мати один лише зіпсований піксель і воно буде визнаватися за даним критерієм сильно зіпсованим.

$$\Delta_{max} = max_{x,y}|f(x,y) - f^{|}(x,\ y)| \quad (1.4)$$

На практиці користуються мірою оцінки якості за критерієм співвідношення сигнал – шум, який визначається за формулою 1.5. Цей критерій аналогічний середньому квадратичному відхиленню, але користуватися ним набагато зручніше завдяки логарифмічному масштабу шкали [7].

$$s = 10\ \log_{10} \frac{255^2 * M * N}{\sum_{x,y}^{M,N} (f(x,y) - f^{|}(x,\ y))^2} \quad (1.5)$$

Найкраще втрати в якості оцінює людський зір. Ущільнення зображення можна вважати відмінним, якщо людина при розгляді ущільненого зображення не зможе відрізнити оригінал від ушіотненого зображення. Але на практиці при ущільненні з втратами на зображення майже завжди виникають спотворення, помітні при порівнянні оригіналу і стисненого зображення.

Високої швидкості ущільнення та відновлення зображення можна досягти за рахунок зменшення кількості та складності обчислень. Висока швидкість ущільнення та відновлення є необхідною при потребі частого редагування зображень, оскільки для редагування ущільнене зображення потрібно спочатку відновити, а потім знову ущільнити. При потребі швидкого перегляду багатьох ущільнених зображень або при передачі по мережі необхідною є лише висока швидкість виконання відновлення. Заради вищезгаданого критерію доводиться жертвувати високим коефіцієнтом ущільнення та низькими втратами якості відновленого зображення .

Масштабування зображень передбачає легку зміну розмірів зображення до розмірів вікна, у якому воно відображається. Це пов’язано з тим, що одні алгоритми ущільнення дозволяють легко масштабувати зображення прямо під час декомпресії, в той час як інші не тільки не дозволяють легко масштабувати, але і збільшують ймовірність появи видимих дефектів після застосування стандартних алгоритмів масштабування до ущільнених зображень. Наприклад, можна навести

приклад «поганого» зображення для алгоритму JPEG – це зображення з досить дрібним регулярним малюнком (піджак у дрібну клітинку). Характер внесених алгоритмом JPEG спотворень такий, що зменшення або збільшення зображення може дати неприємні ефекти [8].

Можливість перегляду зменшеної копії зображення при перегляді лише початку файлу трохи втратила свою актуальність із зростанням швидкості передачі даних по мережі, проте залишаються ще причини, щоб враховувати цю вимогу до алгоритмів ущільнення. При наявності мережевого з'єднання з невисокою швидкістю передача зображення може зайняти великий проміжок часу. Якщо після передачі невеликої частини зображення користувач зможе побачити його зменшену копію, то він зможе при бажанні припинити завантаження, у зв'язку з тим, що вже отримав необхідну інформацію з зображення або вирішив, що завантажувати зображення далі не потрібно. На багатьох Інтернет-сайтах у фотогалереях відображаються зменшені копії зображення. Зазвичай для цього використовується окреме зображення, що займає певну кількість дискового простору. У випадку використання певних алгоритмів ущільнення для відображення зменшеної копії можна використати не окремий файл із зменшеною копією, а початок файлу основного зображення. Це дозволить значно зменшити об'єм дискового простору, необхідного для збереження галереї.

Стійкість до помилок означає локальність порушень в зображенні при псуванні або втраті фрагмента переданого файлу. Дана можливість використовується при передачі повідомлення усім підключеним клієнтам (broadcasting – передача за багатьма адресами) зображень по мережі, тобто в тих випадках, коли неможливо використовувати протокол передачі, щоб повторно клієнту послати запит на сервер для повторної передачі даних у випадку виникнення помилок. Наприклад, якщо передається зображення і виникає помилка при передачі двох-трьох байт, то було б неправильно використовувати алгоритм, у якого збій призводив би до погіршення якості всього зображення, а не лише кількох його пік селів.

Врахування специфічних особливостей зображень дозволяє значно

збільшити коефіцієнт ущільнення при невидимих для людського ока втратах якості у відновленого зображення. Різні алгоритми ущільнення дають різний результат при ущільненні різних видів зображень. Наприклад, алгоритм ущільнення JPEG зумовить виникнення високих втрат якості при ущільненні зображення з великою кількістю дрібних деталей. Тому у даному випадку потрібно використати інший алгоритм ущільнення.

Можливість редагування зображень дозволяє досягти мінімального ступеня погіршення якості зображення при його повторному збереженні після редагування. Багато алгоритмів ущільнення з втратою якості можуть істотно зіпсувати зображення за декілька ітерацій редагування, оскільки при кожній ітерації будуть накопичуватися різноманітні мілкі дефекти зображення.

Невелика вартість апаратної та програмної реалізації висувається до алгоритмів ущільнення у зв'язку з обмеженістю економічних ресурсів. Ці вимоги до алгоритмів ущільнення пред'являють не лише виробники ігрових приставок, але і виробники багатьох інформаційних систем. Так, декомпресор фрактального алгоритму дуже ефективно і коротко реалізується з використанням технології MMX і розпаралелювання обчислень, а стиснення за стандартом CCITT Group 3 легко реалізується апаратно.

1.2 Порівняльний аналіз аналогів

Серед найбільш відомих аналогів можна відзначити такі застосунки:

– RIOT (Radical Image Optimization Tool);

– FileOptimizer;

– Caesium Image Compressor;

– ImageOptim;

RIOT — це безкоштовна Windows-програма для оптимізації зображень з можливістю попереднього перегляду та детального налаштування параметрів стиснення [9] (див. рисунок 1.1).

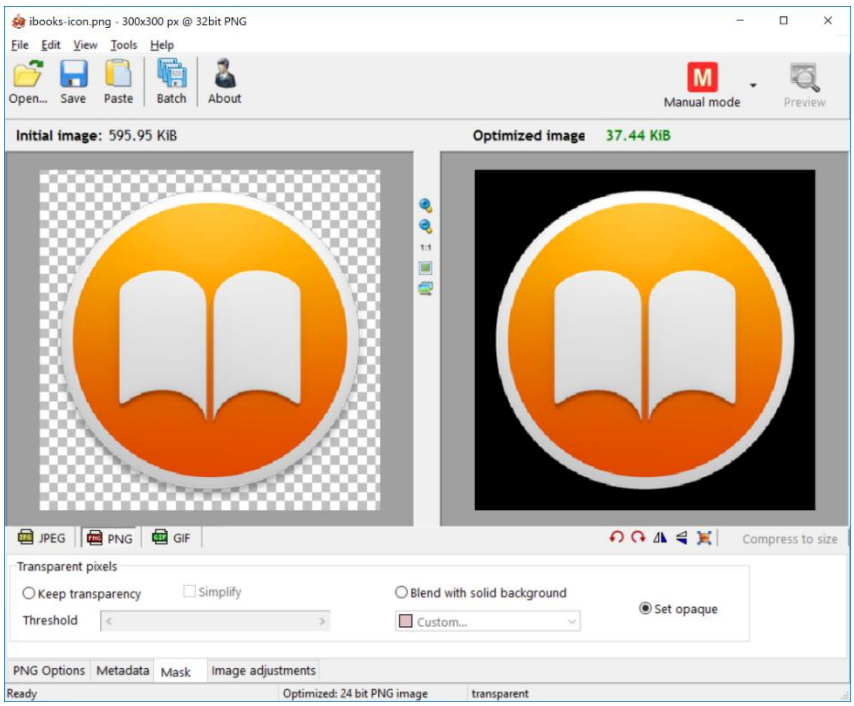


Рисунок 1.1 – Приклад роботи застосунку RIOT

FileOptimizer — це багатофункціональний безкоштовний застосунок для оптимізації різноманітних типів файлів без втрати якості, зокрема графічних (див. рисунок 1.2) [10]. Його головна мета — зменшення розміру файлів без змін у візуальному відображенні, що робить його ідеальним для підготовки зображень до публікації в Інтернеті, електронної пошти або архівування.

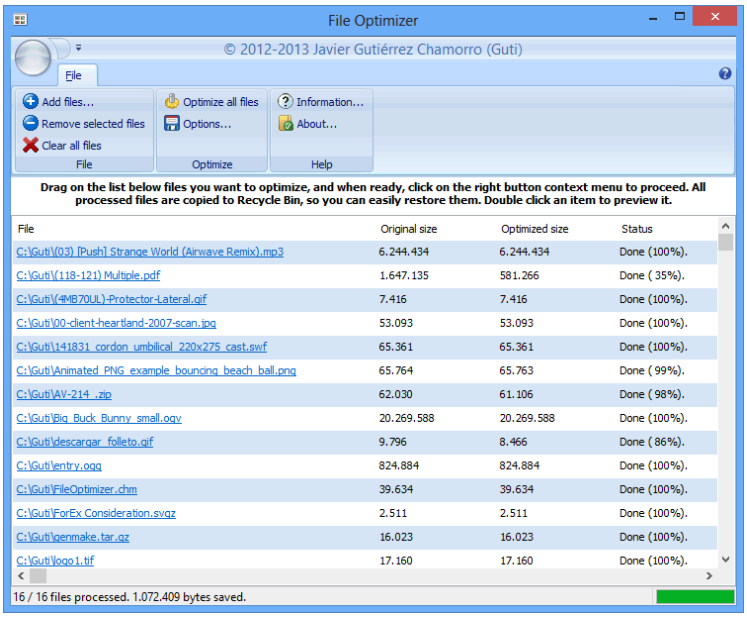


Рисунок 1.2 – Приклад роботи застосунку FileOptimizer

Caesium Image Compressor – це безкоштовна, відкрита та кросплатформна програма, створена для оптимізації зображень (див. рисунок 1.3) [11]. Вона орієнтована на користувачів, яким потрібно зменшити розмір зображень для веб-сайтів, презентацій, електронної пошти або архівування.

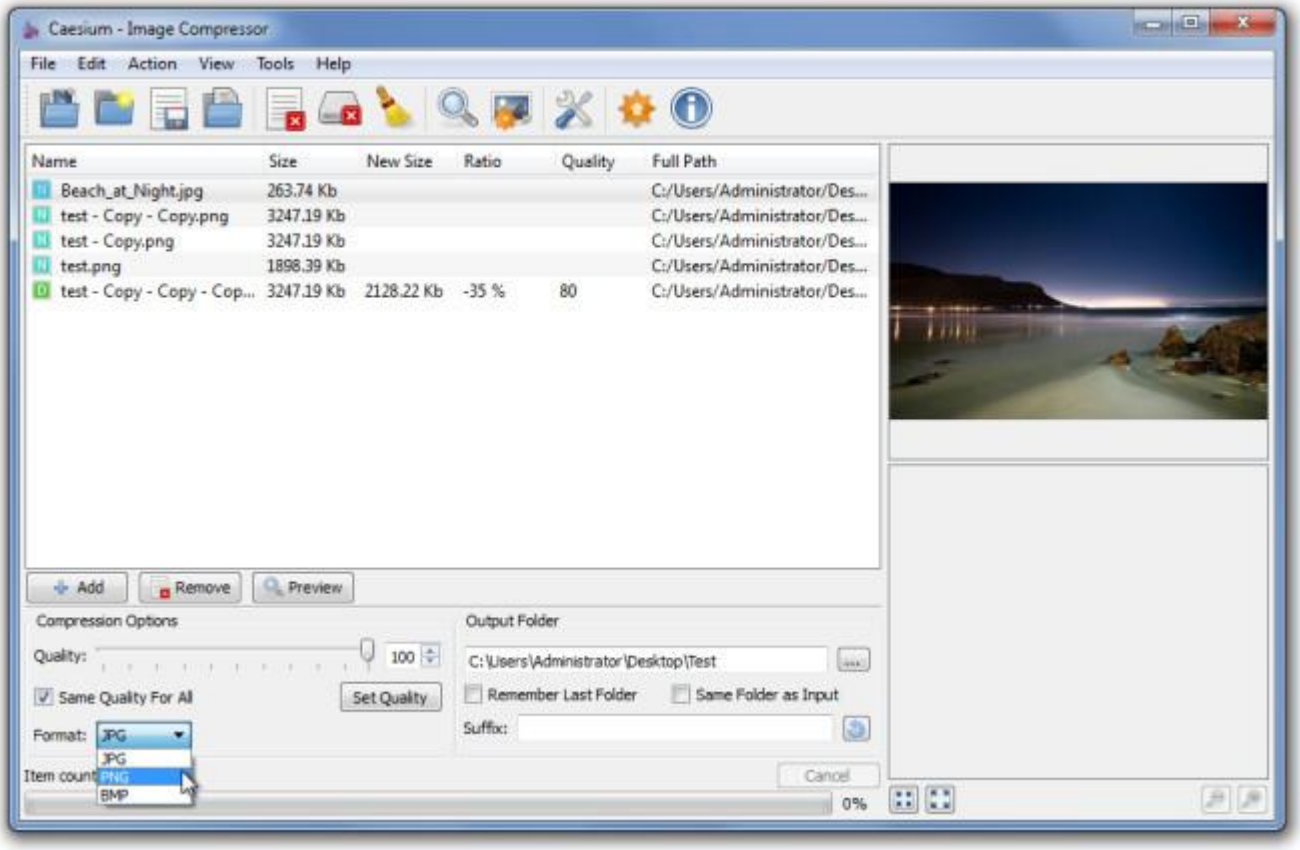


Рисунок 1.3 – Головне вікно Caesium Image Compressor

ImageOptim — це високоякісний, безкоштовний застосунок для macOS, який забезпечує ефективне ущільнення зображень без помітної втрати візуальної якості [12]. Цей інструмент користується великою популярністю серед дизайнерів, веб-розробників, фотографів і мобільних розробників завдяки своїй простоті, стабільності та надзвичайній ефективності (див. рисунок 1.4).

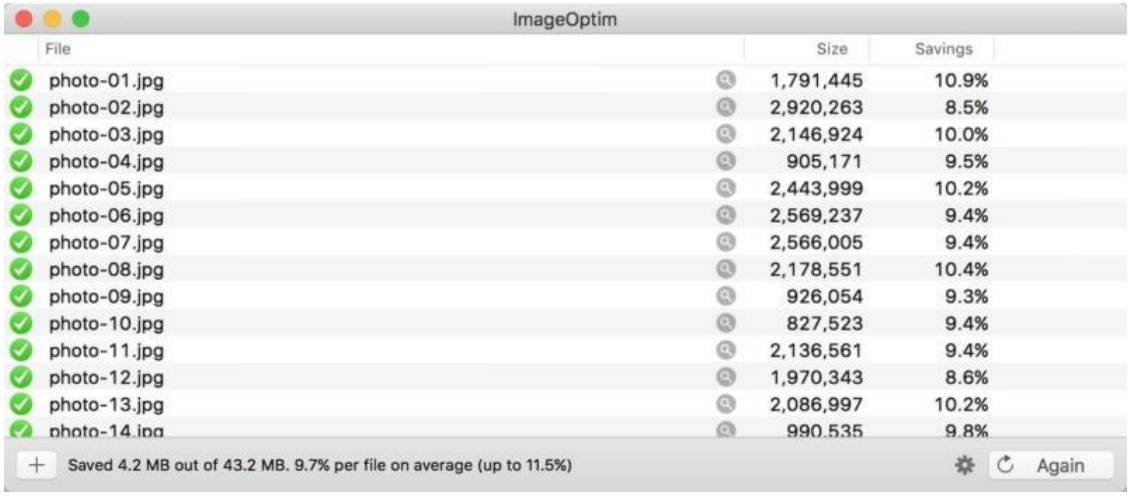


Рисунок 1.4 – Приклад роботи застосунку ImageOptim

Було зроблено порівняльну характеристику розроблюваного продукту та наявних аналогів, результати порівняння зведено в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій	**RIOT**	**File Optimizer**	**CIC**	**ImageOptim**	**Власна розробка**
Власний алгоритм стиснення	-	-	-	-	+
Підтримка рекурсивної обробки	+	+	+	+	+
Адаптивне стиснення	-	-	-	-	+
Точний контроль якості	+	-	+	-	+
Швидкодія при обробці великих обсягів	+	-	+	+	+

Продовження таблиці 1.1

Критерій	**RIOT**	**File Optimizer**	**CIC**	**ImageOptim**	**Власна розробка**
Простота інтерфейсу	+	-	+	+	+
Підтримка популярних форматів	+	+	+	+	+
Видалення метаданих (EXIF)	+	+	+	+	+
Можливість масової обробки	+	+	+	+	+
Гнучкість налаштувань	-	-	-	-	+
Підтримка кількох платформ	-	-	+	-	+
Відкритий вихідний код	+	+	+	+	+
Безкоштовність	+	+	+	+	+
Загальний бал (+)	**9**	**7**	**10**	**9**	**13**

Як видно з таблиці, жоден із наявних застосунків для ущільнення зображень не поєднує всі ключові можливості в одному рішенні: повної підтримки широкого спектра графічних форматів, зручного інтерфейсу, точної настройки рівня стиснення, автоматичної рекурсивної обробки каталогів, видалення метаданих, а також одночасного використання як вбудованих, так і власних алгоритмів. Деякі інструменти, як-от Adobe Photoshop, орієнтовані на професійне редагування, але не спеціалізуються на масовому ущільненні [13]. Інші, такі як RIOT чи Caesium, хоча й мають простий інтерфейс, пропонують обмежену функціональність або працюють лише на певних операційних системах.

Саме тому доцільною є розробка власного програмного застосунку, який би

об'єднав найкращі риси конкурентів і розширив їх можливості за рахунок використання рекурсивного методу ущільнення, що дозволяє гнучко адаптувати алгоритми до різних типів зображень та досягати вищого рівня ефективності стиснення.

Таким чином, створення комплексної та кросплатформної системи ущільнення зображень на основі рекурсивного методу є актуальним напрямком розробки програмного забезпечення. Такий інструмент буде корисним не лише для звичайних користувачів, а й для розробників, дизайнерів і фахівців, що працюють із великими обсягами графічного контенту..

1.3 Класифікація алгоритмів ущільнення зображень

Усі методи ущільнення інформації засновані на тому простому припущенні, що набір даних завжди містить надлишкові елементи.

Ущільнення досягається за рахунок пошуку та кодування надлишкових елементів [14].

Потік даних про зображення має істотну кількість зайвої інформації, яка може бути усунена практично без помітних для людського зору спотворень. При цьому розрізняють два типи надмірності: статистична і візуальна.

Статистична надмірність пов’язана з кореляцією і передбачуваністю даних. Ця надмірність може бути усунена без втрати інформації, вихідні дані при цьому можуть бути повністю відновлені. Найбільш відомі методи ефективного кодування символів засновані на знанні частоти кожного символу, присутнього в повідомленні. Знаючи ці частоти, будують таблицю кодів, що володіє наступними властивостями:

– різні коди можуть мати різну кількість біт;

– коди символів з більшою частотою зустрічі, мають менше біт, ніж коди символів з меншою частотою;

– хоча коди мають різну бітову довжину, вони можуть бути відновлені єдиним чином, тобто коди будуються як префіксні.

Цими властивостями володіє відомий алгоритм Хаффмана.

Візуальна (суб'єктивна) надмірність, яку можна усунути за частковою втратою даних, які мало впливають на якість відтворюваних зображень; це – інформація, яку можна вилучити з зображення, не порушуючи візуального сприйняття якості зображень.

Усунення візуальної надмірності зображень є основним резервом ущільнення зображень. Для оптимізації процесу кодування з метою забезпечення високого коефіцієнта ущільнення необхідно, з одного боку, не зберігати надлишкову інформацію, а з іншого, – не допустити надмірної втрати якості зображення.

До цих пір не існує простої та адекватної моделі візуального сприйняття зображень, придатної для оптимізації їх кодування. Завдання ущільнення зображення складається з двох основних частин: кодування і декодування. Якщо декодуване зображення завжди в точності відповідає кодованому зображенню, то такий алгоритм кодування-декодування називається алгоритмом ущільнення без втрат. Якщо декодоване зображення відрізняється від кодованого, то подібний алгоритм називають алгоритмом ущільнення з втратами.

Існує кілька різних підходів до проблеми ущільнення інформації. Одні мають вельми складну теоретичну математичну базу, інші засновані на властивостях інформаційного потоку і алгоритмічно досить прості. Будь-який спосіб, який реалізує ущільнення даних, призначений для зниження обсягу вихідного потоку інформації допомоги оборотного або необоротного перетворення. Тому всі способи ущільнення можна розділити на дві категорії: зворотнє і незворотнє ущільнення. Методи ущільнення цифрових зображень можна класифікувати за їх основними характеристиками, такими як: точність відновлення, симетричність основного перетворення і тип використовуваного перетворення.

Існують такі методи ущільнення зображення:

1. Ущільнення зображень без втрат:
   - алгоритми Хафмана,
   - алгоритм RLE,

– алгоритм LZW,

– кодування за ступенем новизни;

2. Ущільнення зображеньз втратами:

– фрактальне ущільнення,

– кодування по стандарту JPEG,

– Wavelet-кодування.

Ущільнення без втрат завжди призводить до зниження обсягу вихідного потоку інформації без зміни його інформативності, тобто без втрати інформаційної структури. Більше того, з вихідного потоку, за допомогою алгоритму, який виконує відновлення, можна отримати вхідний.

Ущільнення з втратами – це перетворення вхідного потоку даних, при якому вихідний потік, заснований на певному форматі інформації, представляє досить схожий за зовнішніми характеристиками на вхідних потік об’єкт, проте відрізняється від нього об'ємом. Ступінь подібності вхідного і вихідного потоків визначається ступенем відповідності деяких властивостей об’єктів (тобто ущільненої і неущільненої інформації у відповідності з певним форматом даних), репрезентованої даним потоком інформації.

Саме такі алгоритми використовуються для даних растрових графічних файлів з низьким ступенем повторюваності байтів в потоці. При такому підході використовується властивість структури формату графічного файлу і можливість представити графічну картинку, приблизно схожу за якістю відображення (для сприйняття людським оком), декількома способами. Тому, крім ступеня або величини ущільнення, в таких алгоритмах виникає поняття якості, тому що вихідне зображення в процесі стиснення змінюється. Під якістю можна розуміти ступінь відповідності вихідного і результуючого зображення. Для графічних файлів така відповідність визначається візуально, хоча є і відповідні формалізовані методики і оцінки. Ущільнення з втратами неможливо застосовувати в областях, в яких необхідно мати точну відповідність інформаційної структури вхідного і вихідного потоків.

Методи ущільнення без втрат використовуються в основному в науці та

медицині, коли втрата інформації неприпустима або самі шуми зображення є головною інформацією, наприклад, в системах оцінки якості оптико-електронних систем. Коефіцієнт ущільнення досягається цими методами не більше 1,5 для фотореалістичних зображень. Методи ущільнення з втратами дозволяють отримати істотно більші коефіцієнти ущільнення.

Загальна схема процесу ущільнення зображення з втратами показана на рисунку 1.1.

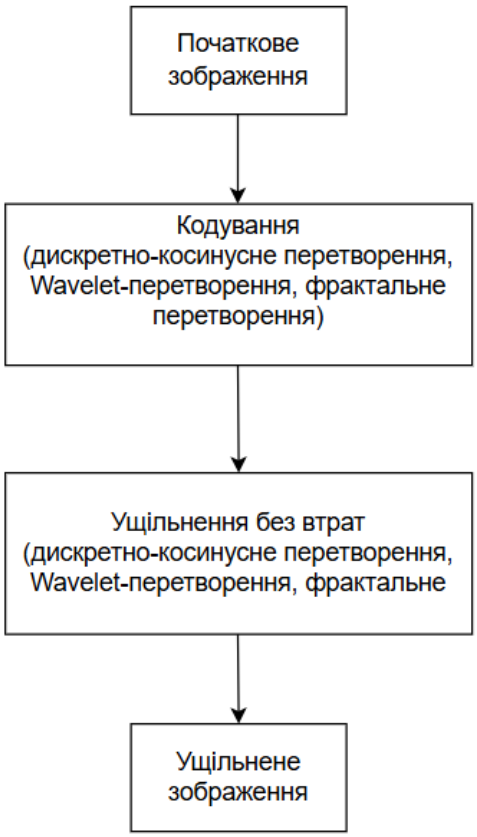


Рисунок 1.1 – Схема ущільнення зображень з втратами

Розглянемо етапи процедури стиснення даних в загальному вигляді. Будь-який метод ущільнення зображень з втратами реалізується у два етапи:

– кодування або первинне ущільнення;

– вторинне ущільнення.

На першому етапі виконується перетворення вихідних даних з однієї форми

подання в іншу. Зокрема, при ущільненні зображень залежно від виду алгоритму ущільнення може бути виконаний перехід від вихідного зображення до наступних видів подання.

На другому етапі компоненти перетворення квантуються і приводяться до вигляду, зручного для статистичного кодування, а потім кодуються. На цьому етапі забезпечується ущільнення інформаційного потоку.

1.4 Алгоритми ущільнення зображень без втрат

Алгоритм Хафмана

Кодування Хаффмана є простим алгоритмом для побудови кодів змінної довжини. Цей популярний алгоритм служить основою багатьох комп'ютерних програм ущільнення текстової та графічної інформації. Деякі з них використовують безпосередньо алгоритм Хаффмана, а інші беруть його в якості однієї із ступенів багаторівневого процесу стиснення. За допомогою методу Хаффмана можна досягти ідеального ущільнення (тобто ущільнення даних до їх ентропії), якщо ймовірності символів дорівнюють від'ємним степеням числа 2. Алгоритм починає будувати кодове дерево знизу вгору, потім ковзає вниз по дереву, щоб побудувати кожен індивідуальний код справа наліво (від наймолодшого біта до найстаршого) [15]. Починаючи з робіт Д.Хаффмана 1952 року народження, цей алгоритм був предметом багатьох досліджень.

Алгоритм починається з складання списку символів алфавіту в порядку зменшення їх ймовірностей. Потім від кореня будується дерево, вузлами якого служать ці символи. Це робиться по кроках, причому на кожному кроці вибираються два символи з найменшими ймовірностями, додаються наверх часткового дерева, видаляються зі списку і замінюються допоміжним символом, що представляє ці два символи. Допоміжному символу приписується ймовірність, що дорівнює сумі ймовірностей, обраних на цьому кроці символів. Коли список скорочується до одного допоміжного символу, що представляє весь алфавіт, дерево оголошується побудованим. Завершується алгоритм спуском по дереву і

побудовою кодів всіх символів.

Алгоритми RLE

Алгоритм RLE надзвичайно простий в реалізації. RLE (Run Length Encoding), або групове кодування, один з найбільш давніх алгоритмів кодування графіки. Його суть полягає в заміні символів, що повторюються, на один цей символ та лічильник повторень. Проблема є в тому, щоб програма, яка виконує ущільнення, при відновленні могла відрізнити в результуючому потоці таку кодовану серію від інших символів. Рішення очевидне – помістити у всі ланцюжки деякі заголовки (наприклад, використати перший біт як ознаку кодування серії). Метод достатньо ефективний для графічних зображень в форматі «байт на піксел» (наприклад, формат PCX використовує кодування RLE) [16].

Алгоритми LZW

Алгоритм LZW відрізняють висока швидкість роботи як при упаковці, так і при розпакуванні, досить скромні вимоги до пам'яті і проста апаратна реалізація. Недолік – низький ступінь ущільнення у порівнянні зі схемою двоступеневого кодування. Алгоритм перетворює потік символів на вході в потік індексів комірок словника на виході існує досить велике сімейство LZW – подібних алгоритмів, що розрізняються, наприклад, методом пошуку повторюваних ланцюжків [17].

Коефіцієнти ущільнення: 1/1000, 1/4, 7/5. Коефіцієнт 1/1000 досягається тільки на одноколірних зображеннях розміром більше 4 Мб. Ситуація, коли алгоритм збільшує зображення, зустрічається вкрай рідко. Ущільнення забезпечується за рахунок однакових ланцюжків в потоці. Алгоритм є симетричним, тобто відновлені після ущільнення дані повністю збігаються з оригіналом, за умови оптимальної реалізації операції пошуку рядка в таблиці. LZW універсальний – саме його варіанти використовуються в звичайних програмах ущільнення даних. Він реалізований у форматах GIF, TIFF і TGA.

Кодування за ступенем новизни

Ідея методу така: нехай алфавіт джерела складається з N символів з номерами 1,2,...,N. Кодер зберігає список символів, які являють собою деяку перес-

тановку алфавіту. Коли на вхід поступає символ С, який має в списку номер “i”, кодер передає код номеру “i”. Потім кодер переставляє символ С на початок списку, збільшуючи на одиницю номери всіх символів, які стоять перед С. Таким чином, більш "популярні" символи будуть тяжіти до початку списку та будуть мати більш короткі коди [18].

В якості коду для кодування за ступенем новизни можна використовувати монотонний код – універсальний код джерела, для якого відомо лише впорядкованість ймовірностей символів. Монотонний код побудований так, що більш близькі до початку списку позиції отримують більш короткі коди. В результаті літери, які часто з'являються та тяжіють до початку списку, будуть мати короткі кодові позначення. Розглянемо стискання інформації методом кодування за ступенем новизни на прикладі.

1.5 Алгоритми ущільнень зображень з втратами

Фрактальне ущільнення

За допомогою фракталів можна ущільнювати великі растрові зображення до частин їхніх нормальних розмірів. Це твердження випливає з теореми Банаха про стискуючі перетворення й є результатом роботи дослідника Технологічного інституту шт. Джорджія Майкла Барнслі [19].

Коротко метод можна описати таким чином. Зображення кодується кількома простими перетвореннями (в нашому випадкові афінними), тобто визначається коефіцієнтами цих перетворень (в нашому випадкові: A, B, C, D, E та F).

Наприклад, закодувавши якесь зображення двома афінними перетвореннями, ми однозначно визначаємо його за допомогою 12 коефіцієнтів. Якщо тепер задати яку-небудь початкову точку (наприклад, X = 0, Y = 0) та запустити ітераційний процес, то ми після першої ітерації отримаємо дві точки, після другої — чотири, після третьої — вісім і т. д. Через кілька десятків ітерацій сукупність отриманих точок описуватиме закодоване зображення. Але проблема полягає в тому, що дуже важко знайти коефіцієнти перетворень, які кодували б

довільне зображення [20].

Перевагою фрактального ущільнення є високий коефіцієнт ущільнення. Основний недолік – дуже низька швидкодія виконання ущільнення та відновлення зображень.

Кодування по стандарту JPEG

Алгоритм JPEG найбільшою мірою придатний для ущільнення фотографій і картин, які містять реалістичні сцени з плавними переходами яскравості і кольору. Найбільшого поширення JPEG отримав у цифровій фотографії і для зберігання і передачі зображень з використанням мережі Інтернет [21] .

З іншого боку, JPEG малопридатний для ущільнення креслень, текстової і знакової графіки, де різкий контраст між сусідніми пікселями приводить до появи помітних дефектів. Такі зображення доцільно зберігати у форматах без втрат, таких як TIFF , GIF або PNG.

JPEG (як і інші методи ущільнення) не підходить для ущільнення зображень при багатоступінчастій обробці, так як спотворення в зображення будуть вноситися щоразу при збереженні проміжних результатів обробки.

JPEG не повинен використовуватися і в тих випадках, коли неприпустимі навіть мінімальні втрати, наприклад, при ущільненні астрономічних або медичних зображень. У таких випадках може бути рекомендований передбачений стандартом JPEG режим стиснення Lossless JPEG) або стандарт стиснення JPEG-LS.

Процес виконання кодування-декодування відповідно до стандарту JPEG показаний на рисунку 1.2.

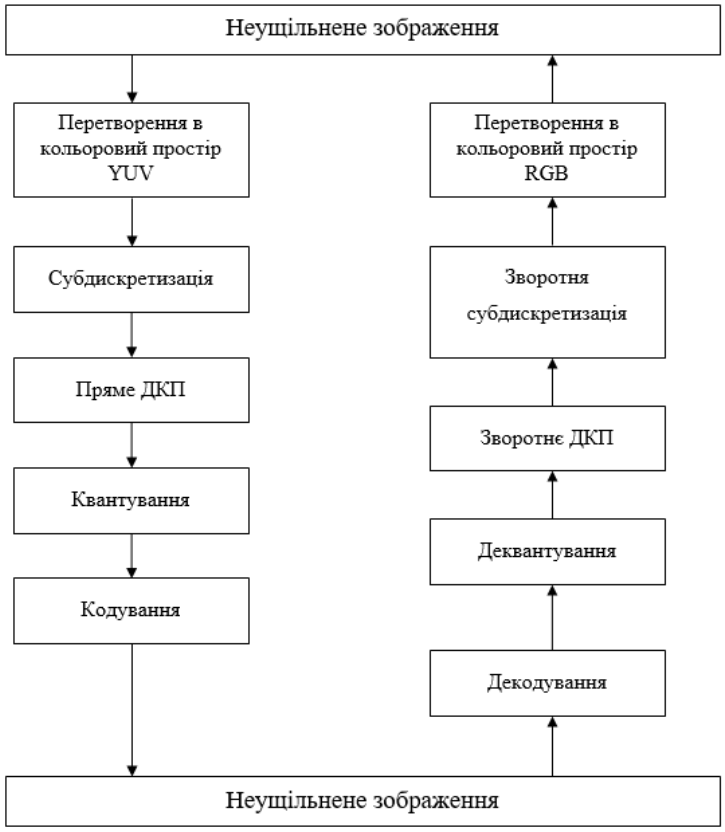


Рисунок 1.2 – Схема виконання кодування-декодування відповідно до стандарту JPEG

До недоліків ущільнення за стандартом JPEG слід віднести появу на відновлених зображеннях при високих ступенях стиснення характерних дефектів: зображення розсипається на блоки розміром 8x8 пікселів (цей ефект особливо помітний на областях зображення з плавними змінами яскравості), в областях з високою просторовою частотою (наприклад, на контрастних контурах і границях зображення) виникають дефекти у вигляді шумових ареалів. Слід зазначити, що стандарт JPEG (ISO / IEC 10918-1, Annex K, п. K.8) передбачає використання спеціальних фільтрів для придушення блокових дефектів, але на практиці подібні фільтри, незважаючи на їх високу ефективність, практично не використовуються. Ще одним недоліком є висока складність обчислень при виконанні дискретно-косинусного перетворення. Однак, незважаючи на недоліки, JPEG отримав дуже широке поширення через досить високий (щодо існування під час його появи альтернатив) коефіцієнт ущільнення, підтримки ущільнення повнокольорових

зображень.

Wavelet-кодування

Алгоритм орієнтований на ущільнення кольорових і чорно-білих зображень з плавними переходами, ідеальний для картинок типу рентгенівських фотографій. Коефіцієнт ущільнення варіюється в межах 5-100. При великих коефіцієнтах ущільнення на різких границях, особливо діагональних, можливі спотворення [21].

Основна ідея алгоритму полягає в тому, що зберігаються різниці між середніми значеннями сусідніх блоків зображення, які звичайно приймають значення, близькі до нуля.

До достоїнств цього алгоритму можна віднести те, що він дуже легко дозволяє реалізувати можливість поступового «прояву» зображення при передачі зображення по мережі. Крім того, оскільки на початку зображення ми фактично зберігаємо його зменшену копію, спрощується показ «огрубленого» зображення по заголовку.

Ущільнення базується на пірамідальному S-перетворенні, яке може використовуватись для ущільнення фотореалістичних зображень. Ущільнення виконується за два кроки: перший – S-перетворення початкового зображення; другий – перетворені дані кодуються одним з статистичних методів. Обидві операції зворотні, що дозволяє відновити початкове зображення. Однак для отримання великих коефіцієнтів ущільнення необхідно зниження точності представлення компонент зображення, отриманих в результаті виконання S-перетворення, але так, щоб спотворення не були візуально помітні.

На відміну від JPEG і фрактального алгоритму, S-перетворення не оперує з блоками, наприклад, 8x8 пікселів, а оперує блоками 2x2, 4x4, 8x8 і т.д. Однак за рахунок того, що коефіцієнти для цих блоків ми зберігаємо незалежно, ми можемо досить легко уникнути дроблення зображення на «мозаїчні» квадрати.

Головними перевагами алгоритму ущільнення зображень з використанням Wavelet-перетворень є висока швидкодія виконання операцій кодування та декодування зображень і та можливість модифікації алгоритму для можливості

виконувати повторне ущільнення відновленого зображення майже без втрат.

При використанні алгоритму на основі Wavelet-перетворень коефіцієнт ущільнення можна також збільшити за рахунок зменшення надлишковості зображень.

Розрізняють два основних види надлишковості:

- статистична надлишковість;
- фізіологічна надлишковість.

Перша пов’язана з тим що будь-які величини, отримані із зображення, не є випадковими. Сусідні відліки часто мають подібні значення яскравості, в чому проявляється важлива властивість їх просторової кореляції. Якщо відповідним чином використати цю властивість, то можна значно зменшити число біт для подання зображення в цифровій формі.

Фізіологічна надлишковість пов'язана з тією частиною інформації, яка не сприймається оком людини. Скорочення фізіологічної надлишковості в значній мірі скорочує і статистичну надлишковість і навпаки.

На сучасному етапі розвитку технологій досить часто використовується автоматичний аналіз зображень. Прикладами такого аналізу може бути розпізнавання текстів та образів, пошук зображення у базі даних, порівняння зображень та багато інших. При усуненні фізіологічної надлишковості обов’язково будуть виникати деякі втрати. Для зорового сприйняття людини ці втрати будуть непомітними, але при використанні автоматичного аналізу зображень вони можуть спричинити певні помилки у отриманих результатах. Тому зменшувати фізіологічну надлишковість зображень не рекомендується.

Статистичну надлишковість можна усунути використовуючи алгоритми ущільнення без втрат. Проте звичайні фотореалістичні не ущільненні зображення у більшості випадків не мають великої статистичної надлишковості. Це викликано тим, що при використані колірної моделі RGB фото реалістичні зображення не мають великої статистичної надлишковості. Зміна колірної моделі у деяких випадках дозволяє досягти збільшення статистичної надлишковості.

Особливістю фотореалістичних зображень є те, що багато пікселів мають

однакову яскравість. Отже, кращого коефіцієнта ущільнення дозволить досягти колірна система, яка окремо зберігає компоненту яскравості. Однією з таких систем є система YUV (Y – яскравітсь, U та V різницеві компоненти кольору).

Таким чином для покращення коефіцієнта ущільнення при використанні алгоритму ущільнення зображень на основі Wavelet-перетворень краще використовувати колірну модель YUV [22].

1.6 Висновки

У першому розділі було розглянуто сучасний стан програмних засобів для ущільнення зображень. Проведено аналіз низки популярних рішень, серед яких Adobe Photoshop, RIOT (Radical Image Optimization Tool), FileOptimizer, Caesium Image Compressor та ImageOptim. Кожен із цих застосунків має свої переваги та обмеження, зокрема щодо підтримуваних форматів, ступеня стиснення, можливості пакетної обробки, збереження якості та відкритості коду.

На основі проведеного аналізу було сформульовано вимоги до розробки власного програмного забезпечення для стиснення зображень із використанням рекурсивного методу. Основними завданнями визначено реалізацію модулів рекурсивного поділу зображень, побудову ефективного алгоритму локального ущільнення, створення зручного графічного інтерфейсу, а також забезпечення високого рівня стиснення без значної втрати якості.

Таким чином, було доведено доцільність створення власного застосунку, що поєднує сильні сторони існуючих аналогів і долає їхні ключові недоліки, особливо в аспекті інтелектуального рекурсивного аналізу структури зображення. Визначені завдання та обґрунтований підхід сформували основу для реалізації бакалаврської кваліфікаційної роботи.

2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМИ

2.1 Підхід до ущільнення зображень на основі Wavelet-перетворень

Wavelet-перетворення – це математичні функції, що дозволяють аналізувати різні частотні компоненти даних; це сімейство функцій, які локальні в часі і по частоті і в яких всі функції виходять з однієї за допомогою її зміщення і розтягування по осі часу.

У галузі ущільнення зображень під Wavelet-перетвореннями розуміють велику кількість базисних функцій, які формують компактний опис зображення. Аналіз зображень, побудованих на таких базисах, здійснюється за двома змінними – масштабом і зміщенням. Це дозволяє розділити великі й дрібні деталі зображень, одночасно локалізуючи їх на часовій шкалі. Таким чином, враховується важлива для практики обставина: протяжні об’єкти, даних які аналізуються, лежать в низькочастотній області спектру, а короткі – в високочастотній.

Безсумнівною перевагою застосування цих перетворень є можливість представляти сигнали, які швидко змінюються, в компактній формі. Подібно швидкому перетворенню Фур’є, яке широко використовується, Wavelet-перетворення зворотні і можуть служити інструментом аналізу характеристик сигналів (спектральний аналіз та ін.) На відміну від перетворення Фур’є (де базисними функціями є синус та косинус), Wavelet-перетворення формують батьківські функції більш складної форми. Їх вихідний набір забезпечує отримання нескінченного числа нових форм функцій [23]. Оскільки Wavelet-функції обмежені в просторі, за їх допомогою можна локалізувати просторовий об’єкт з високим ступенем точності.

Неперервне Wavelet-перетворення одномірного сигналу визначається за формулою 2.1.

$$W(a,x) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} f(t) w\left(\frac{t-x}{a}\right) dt \qquad (2.1)$$

де a – масштаб, x – координата, $f(t)$ – початкова функція, t – час.

Результатом перетворення відповідно до формули 2.1 буде функція, яка залежить від двох змінних: координати – x та масштабу – a. Функція $w(x)$ фактично є базисною. На кожному масштабі початковий базис w розтягується по горизонталі і зжимається по вертикалі. Потім він зсувається в точку x, яка досліджується функцією, і згортається разом з нею .

Початкова функція $f(t)$ може бути відновлена за допомогою формули 2.2.

$$f(t) = \frac{1}{C_x} \iint_{-\infty}^{+\infty} \frac{1}{\sqrt{a}} W(a,x) w\left(\frac{t-x}{a}\right) \frac{dadx}{a^2} \tag{2.2}$$

де C_x – норма базисної функції w, яка визначається за формулою 2.3.

$$C_x = \int_0^{+\infty} \frac{\tilde{w}(v)^2}{v} dv \tag{2.3}$$

На кожному етапі Wavelet-перетворення сплеск утворює вікно, вирізаючи з сигналу частину, яка попадає в нього [24]. Значення сигналу, які не потрапили у вікно, не будуть сильно впливати на результат. Таким чином, з'являється можливість локалізувати особливості сигналу. Найпростішим сплеском є функція, на основі якої будується базис Хаара (формула 2.4).

$$f(x) = \begin{cases} 1: x \in [-1,0) \\ -1: x \in [0,1) \\ 0: x \in [-1,1] \end{cases} \tag{2.4}$$

Неперервний Wavelet-аналіз дозволяє отримати велику кількість інформації про сигнал, але разом з тим вимагає багато обчислень. Дії, по обчислювальним затратам еквіваленті перетворенню Фур'є, виконуються для кожного елемента послідовності.

При ущільненні зображення використовуються дискретні Wavelet-перетворення, у яких на відміну від аналізу Фур'є і неперервного Wavelet-аналізу сплески визначені лише на частині області задання аргумента і можуть розглядатися як репліки єдиної базової функції, які відрізняються по масштабу і місцеположенню. Репліка ще називається материнською або породжуючою функцією.

2.2 Розробка метода ущільнення зображень на основі S-перетворення

Приклад схеми пірамідального S-перетворення одномірного сигналу з використанням перетворення Хаара наведено на рис. 2.1. На рисунку 2.2 наведено схему для відновлення закодованого сигналу. Відліки низько- (фільтр F1) і високочастотної (фільтр F2) компонент одержують відповідно до формули 2.5 [24].

$$\begin{aligned} L(n) &= [C(2*n) + C(2*n+1)]/2 \\ H(n) &= [C(2*n) - C(2*n+1)]/2 \end{aligned} \qquad (2.5)$$

де n= 0, 1... (N/2 – 1), C [2*n], C [2*n + 1] – відліки вхідного цифрового сигналу, а L[n], H[n] – низькочастотні і високочастотні коефіцієнти перетворення Хаара (рис. 3.1), тобто L[n] – відліки на виході низькочастотного фільтра F1, H[n] – відліки на виході високочастотного фільтра F2.

До вихідного сигналу фільтрів F1 та F2 знову застосовуються дані формули, утворюючи у такий спосіб піраміду. Ясно, що чим більша кількість компонент, тим більший очікуваний коефіцієнт ущільнення [5].

Перевагою даного представлення вихідного зображення є те, що дисперсія H[n] менша, ніж для C [n].

Зворотнє перетворення Хаара обчислюється за формулою 2.6.

$$\begin{aligned} C(2*n) &= L(n) + H(n), \\ C(2*n+1) &= L(n) - H(n) \end{aligned}. \qquad (2.6)$$

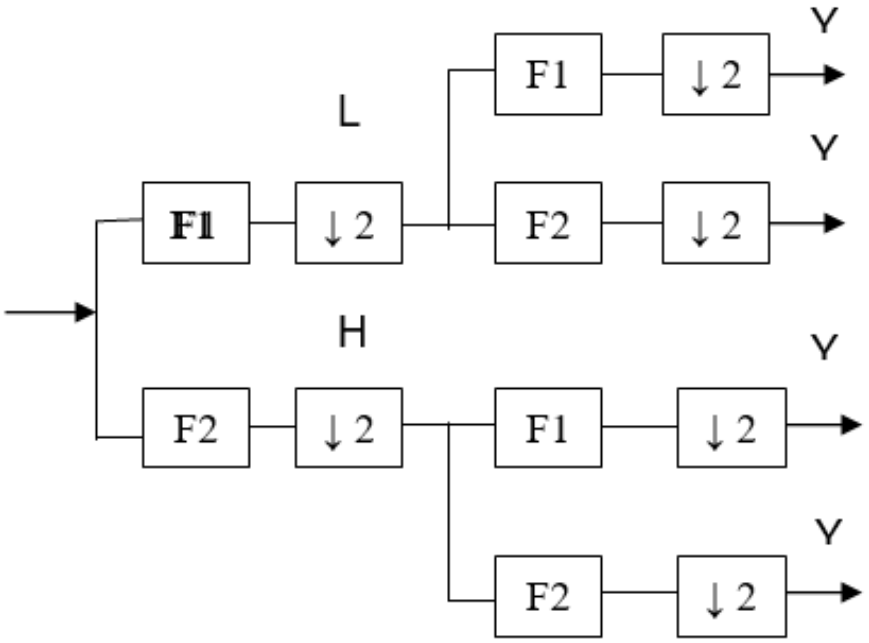


Рисунок 2.1 – Пряме перетворення Хаара

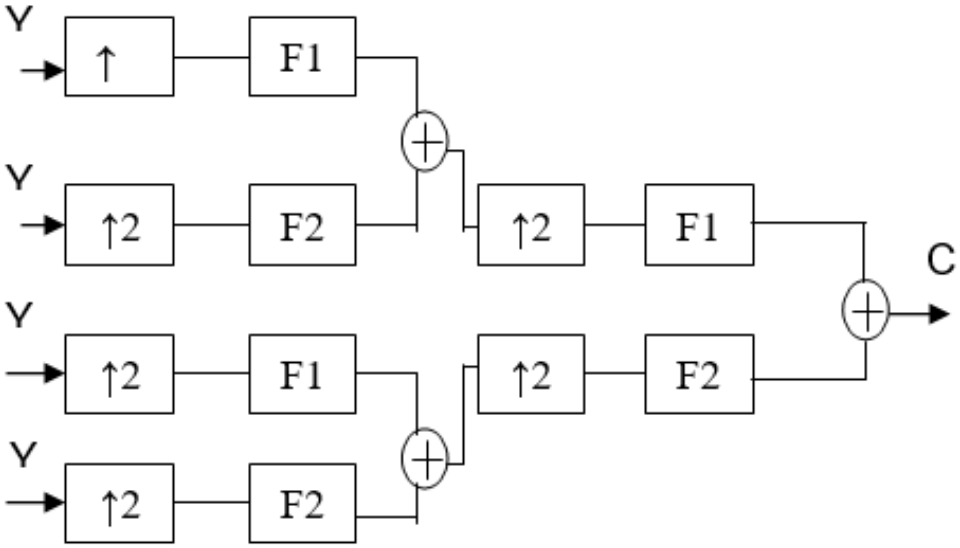
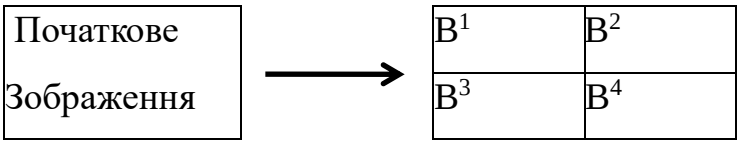


Рисунок 2.2 – Зворотнє перетворення Хаара

Обчислення для двовимірних даних реалізується аналогічно. Для квадрата з розмірами 2x2 і значеннями елементів $a_{2i,2j}$, $a_{2i+1,\,2j}$, $a_{2i,\,2j+1}$, $a_{2i+1,2j+1}$ отримаємо

$$
\begin{aligned}
b^1_{i,j} &= (a_{2i,2j} + a_{2i+1,2j} + a_{2i,2j+1} + a_{2i+1,2j+1})/4 \\
b^2_{i,j} &= (a_{2i,2j} + a_{2i+1,2j} - a_{2i,2j+1} - a_{2i+1,2j+1})/4 \\
b^3_{i,j} &= (a_{2i,2j} - a_{2i+1,2j} + a_{2i,2j+1} - a_{2i+1,2j+1})/4 \\
b^4_{i,j} &= (a_{2i,2j} - a_{2i+1,2j} - a_{2i,2j+1} + a_{2i+1,2j+1})/4
\end{aligned}
\qquad (3.3)
$$

Використовуючи ці формули, наприклад, для зображення з розмірами 512x512 пікселів після першого перетворення одержимо 4 матриці розміром 256x256 елементів:



У першій, як легко догадатися, буде зберігатися зменшена копія зображення, у другій - усереднені різниці пар значень пікселів по горизонталі, У третій — усереднені різниці пар значень пікселів по вертикалі, у четвертій - усереднені різниці значень пікселів по діагоналі.

За аналогією, ми можемо повторити наше перетворення й одержати замість першої матриці 4 матриці розміром 128x128. Повторивши наше перетворення втретє, ми одержимо у підсумку: 4 матриці 64x64, 3 матриці 128x128 і 3 матриці 256x256. Збільшення кількості компонент підвищує коефіцієнт ущільнення. На практиці, при записі у файл, значення $b^4_{i,j}$ звичайно не враховують, відразу одержуючи виграш приблизно на третину розміру файлу.

Загальна структура кодування наведена на рис. 2.3. Вхідні дані обробляються фільтрами S-перетворення, які формують компоненти зображення. Адаптивний квантувач призначений для квантування значень відліків компонент з урахуванням особливостей зорового сприйняття. Процес виконання квантування зв'язаний з втратами інформації, однак візуальна якість відновленого зображення не повинна відрізнятися від вихідного.

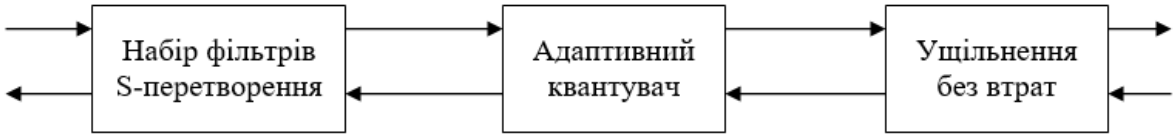


Рисунок 2.3 – Загальна структура кодування

2.3 Розробка алгоритму роботи системи

Програма для ущільнення зображень з використанням рекурсивного методу починає свою роботу, коли користувач запускає застосунок. Спочатку програма надає можливість вибрати зображення для обробки. Після вибору зображення, програма перевіряє його формат, щоб переконатися, що він підтримується (JPEG, PNG, BMP тощо), та відображає попередній перегляд зображення.

Далі користувач може налаштувати параметри ущільнення зображення. Це може включати вибір рівня рекурсії (глибина розбиття зображення на блоки), ступінь компресії (високий, середній, низький) та формат, в якому буде збережено результат. Після вибору налаштувань програма починає процес ущільнення.

Алгоритм ущільнення полягає в тому, що зображення розбивається на менші блоки (наприклад, квадратні ділянки). Програма рекурсивно обробляє кожен блок, оцінюючи його складність. Якщо блок має просту структуру (наприклад, однакові кольори або прості градієнти), програма застосовує більше ущільнення, зменшуючи обсяг даних для цього блоку. Якщо блок містить складні деталі, зображення цього блоку зберігається з мінімальними змінами, щоб зберегти якість.

Після того як всі блоки були ущільнені, програма формує з них фінальне зображення. Це стислий файл, який можна переглянути в попередньому вигляді, щоб користувач міг оцінити результат. Якщо результат влаштовує, програма пропонує зберегти ущільнене зображення. Користувач вибирає місце для збереження файлу та формат, в якому він буде збережений.

Рекурсиве перетворення виконується згідно виразів (2.5). Але якщо один раз застосувати цей вираз, то отримаємо лише 4 компоненти, які не забезпечують достатній коефіцієнт ущільнення для ряду застосувань. Тому при необхідності

необхідно ще раз застосувати двовимірне рекурсивне перетворення, але уже тільки до компоненти B^1, що і відображає спрощена граф-схема алгоритму виконання рекурсивного перетворення на рис. 2.4. Тоді ми зможемо отримати розклад початкового зображення на 7 компонент. Зрозуміло, що користувач повинен мати змогу вибирати 4 або 7 компонент.

Цей алгоритм забезпечує ефективну роботу з зображеннями, гарантуючи оптимальне ущільнення без значної втрати якості, та створює комфортні умови для користувача під час налаштування параметрів та збереження результату.

Перетворення само по собі не ущільнює зображення. Досягнення високих коефіцієнтів ущільнення зображень досягається за рахунок втрат інформації на етапі квантування. Зрозуміло, що ці втрати повинні бути не помітні у відновленому зображенні. Щодо квантування різницевих компонент (B^2, B^3, B^4), то до них можуть бути застосовані логарифмічні шкали квантування (додаток Г), що може значно збільшити коефіцієнт ущільнення.

2.5 Розробка схем та алгоритмів розкладу зображення на 7, 10

У основі розкладу зображення на будь-яку кількість компонент лежить алгоритм та схема розкладу на 4 компоненти. Тобто за допомогою методу розкладу зображення на 4 компоненти можна розкласти зображення на будь-яку кількість компонент. Такий метод обов’язково повинен виконувати кодування певного вікна на зображенні. Для такого методу вхідними параметрами є координати верхнього лівого кута вікна (Xmin, Ymin) та нижнього правого (Xmax, Ymax) і зображення, яке розкладається (Img). Такі ж самі вхідні параметри має кожен метод, який виконує розклад зображення на будь-яку іншу кількість компонент. Коефіцієнт *s* у цих методах показує відношення прощі зображення до площі вікна.

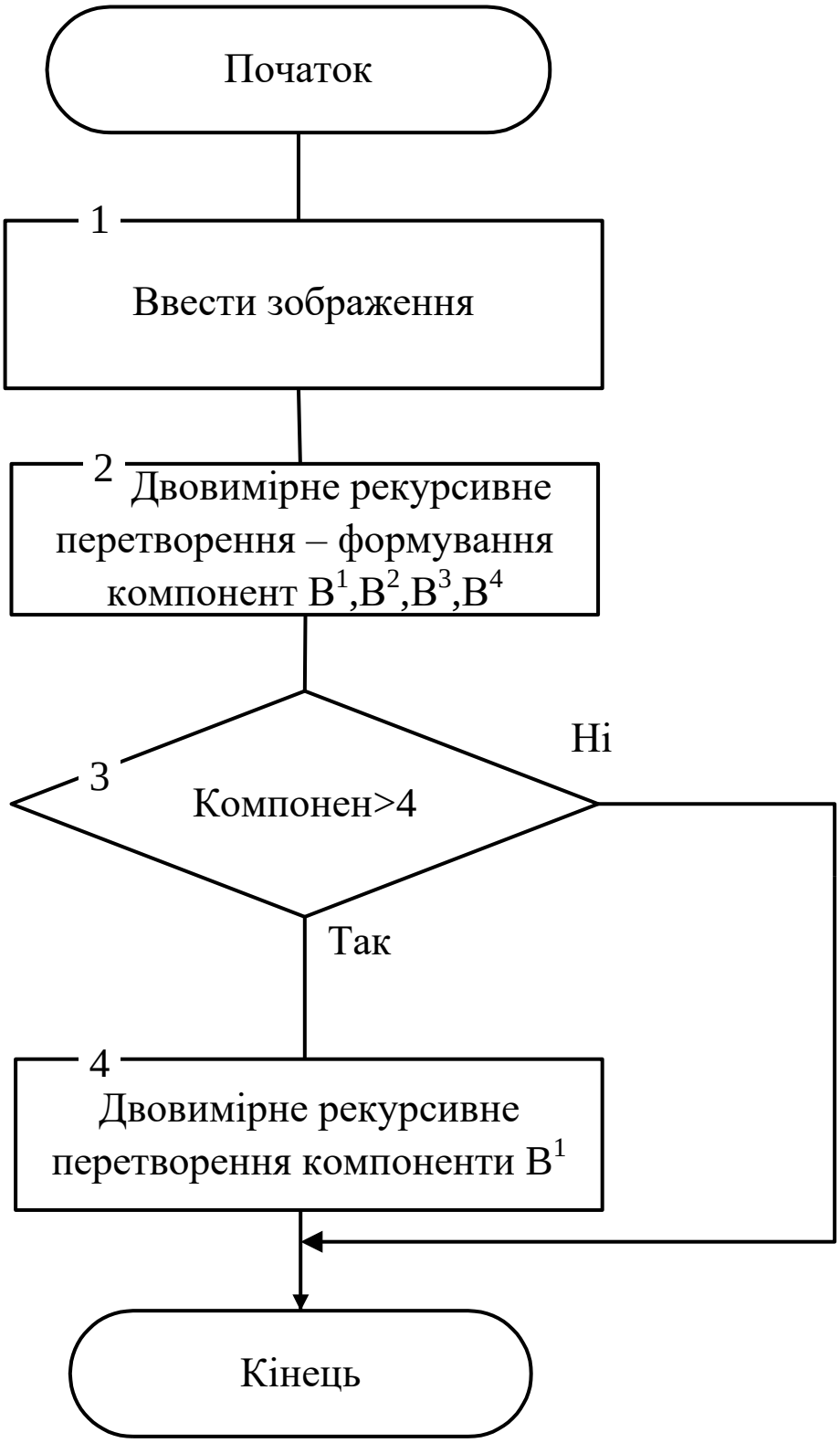


Рисунок 2.4 – Спрощена граф-схема виконання рекурсивного

Семикомпонентний розклад зображення відбувається відповідно до схеми, зображеної на рисунку 2.5. Блок-схему алгоритму методу Comp7 наведено на рисунку 2.6. Коефіцієнт *s* показує відношення площі зображення до площі вікна.

Як видно зі схеми розкладу, зображення, що обробляється, спочатку розкладається на 4 компоненти. Потім компонента B^1 розкладається ще на 4 компоненти. У результаті зображення розкладається на 7 компонент. Такий алгоритм розкладу можна реалізувати за допомогою рекурсії. На блок-схемі алгоритму, зображеній на рис. 2.5, блок №3 алгоритму здійснює рекурсивний виклик процедури. Блок №1 містить умову для виходу з рекурсії. Рекурсія закінчується, коли розмір вікна дорівнює 1/4 розміру зображення.

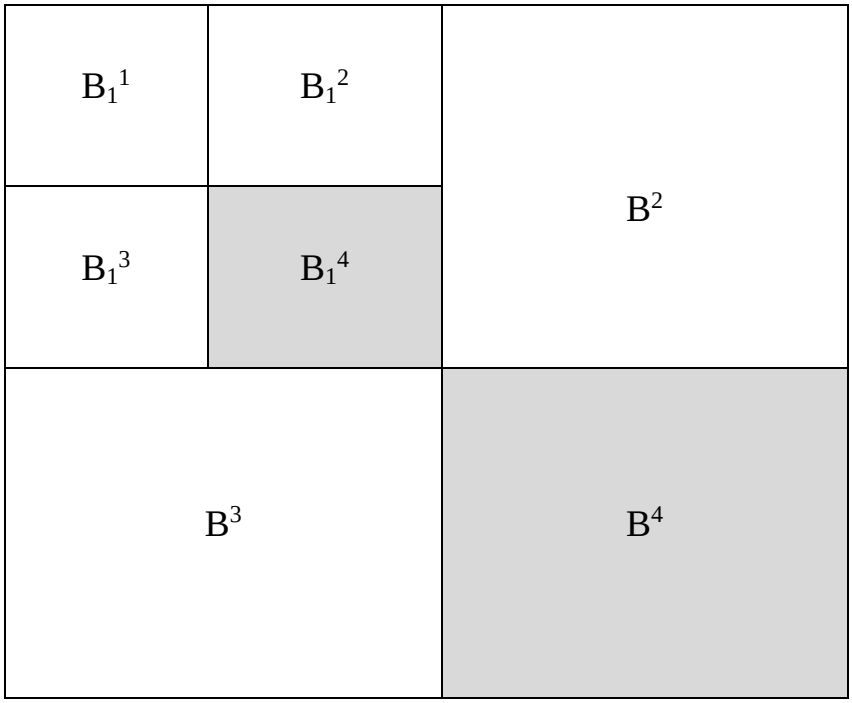


Рисунок 2.5 – Схема розкладу зображення на 7 компонент

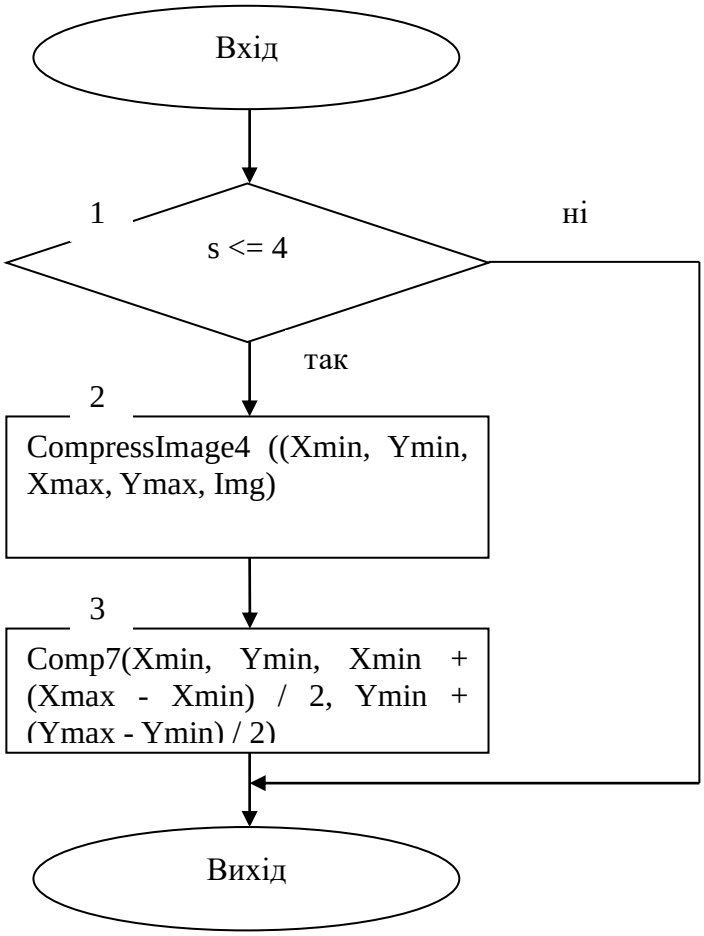


Рисунок 2.6 – Блок-схема алгоритму методу Comp7

На рисунку 2.7 наведено схему розкладу зображення на 10 компонент. Блок-схему алгоритму методу Comp10, який виконує розклад зображення на 10 компонент, наведено на рис. 2.8.

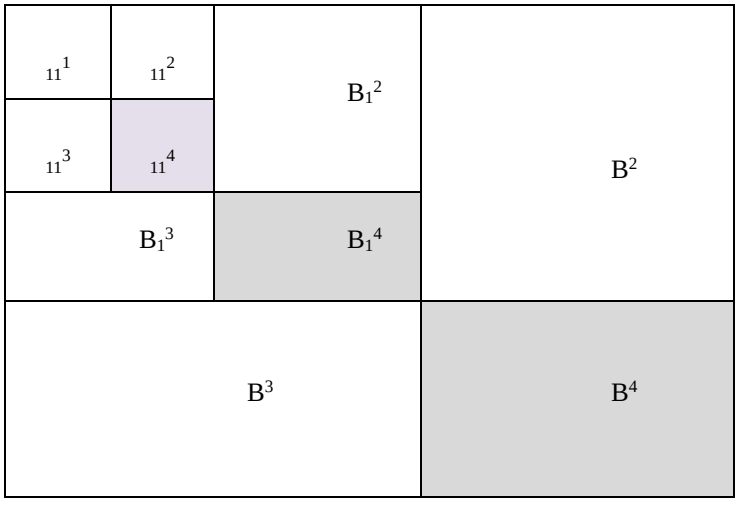


Рисунок 2.7 – Схема розкладу зображення на 10 компонент

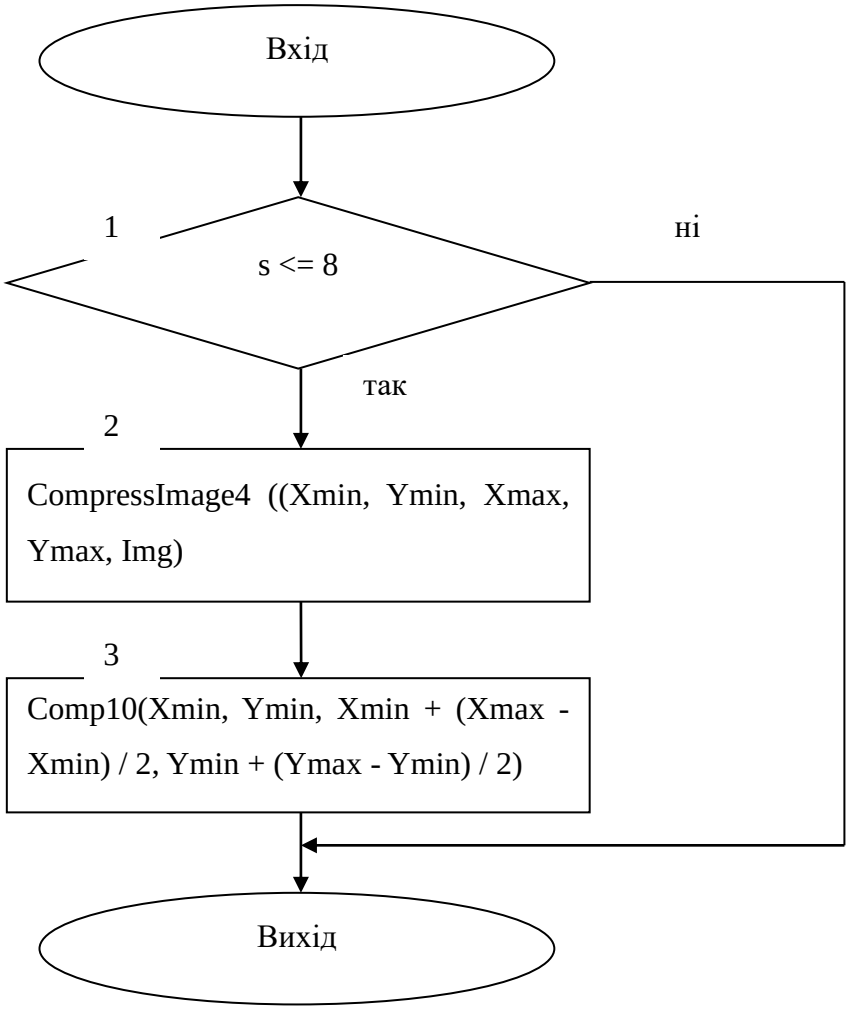


Рисунок 2.8 – Блок-схема алгоритму методу Comp10

2.6 Розробка алгоритму кодування з частковим зберіганням компонент кольору U та V

При використанні кольорової системи YUV при кодуванні можна зберігати лише кожний другий байт кожної з компонент. Звичайно, відновлене зображення буде мати великі втрати, але для людського ока ці втрати будуть непомітними. При кодуванні кожна з компонент кодується окремо.

На рисунку 2.9 наведено приклад блоку U` розміром m на n, який містить компоненту зображення U.

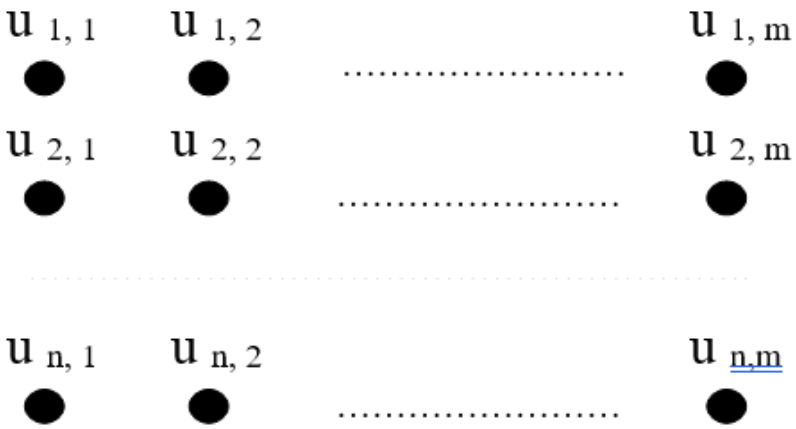


Рисунок 2.9 – Приклад блоку U`, для зберігання компоненти U

На рисунку 2.17 наведено схему розробленого алгоритму кодування з частковим зберіганням компонент U та V, на прикладі блоку U`. Після кодування з блоку U` буде отримано закодований блок S`.

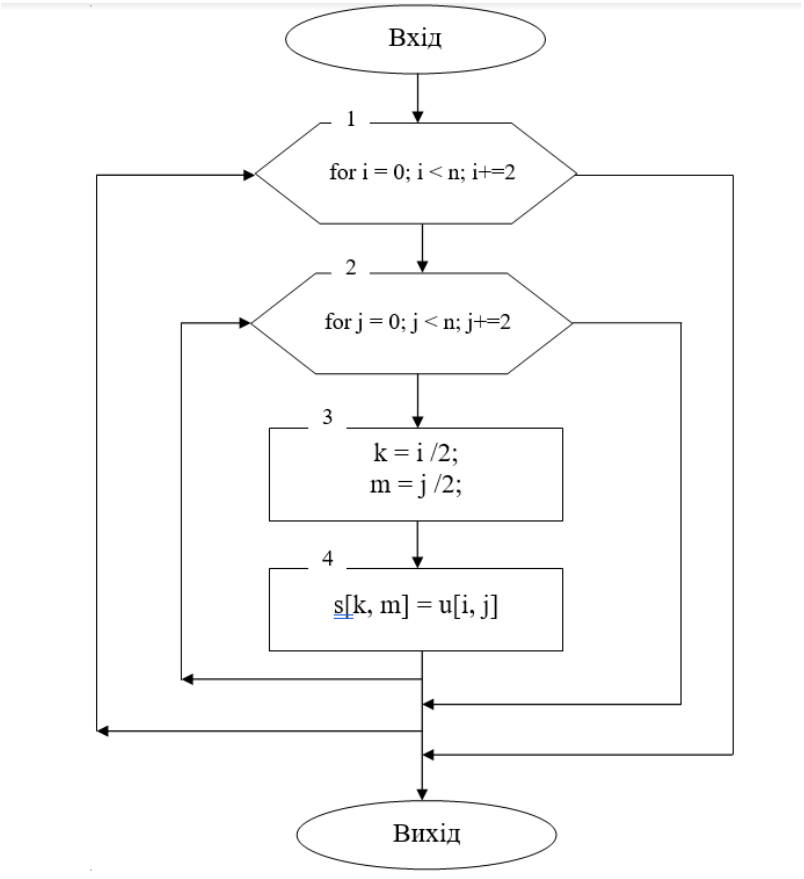


Рисунок 2.17 – Алгоритм кодування з частковим зберіганням компонент U та V для блоку U`

Після кодування зберігається лише 50% інформації. Використовувати наведений алгоритм можна лише до компонент U та V. Застосовувати алгоритм до компоненти Y не рекомендується, тому що дефекти зображення будуть сильно помітними для ока людини. Компоненти U та V при зберіганні займають 66,6% пам’яті. Таким чином при застосуванні розробленого алгоритму до компонент U та V можна ущільнити зображення на 33,3%.

На рисунку 2.18 наведено схему алгоритму декодування закодованого блоку S`. Розмір декодованого блоку дорівнює розміру блоку, який містить компоненту яскравості Y.

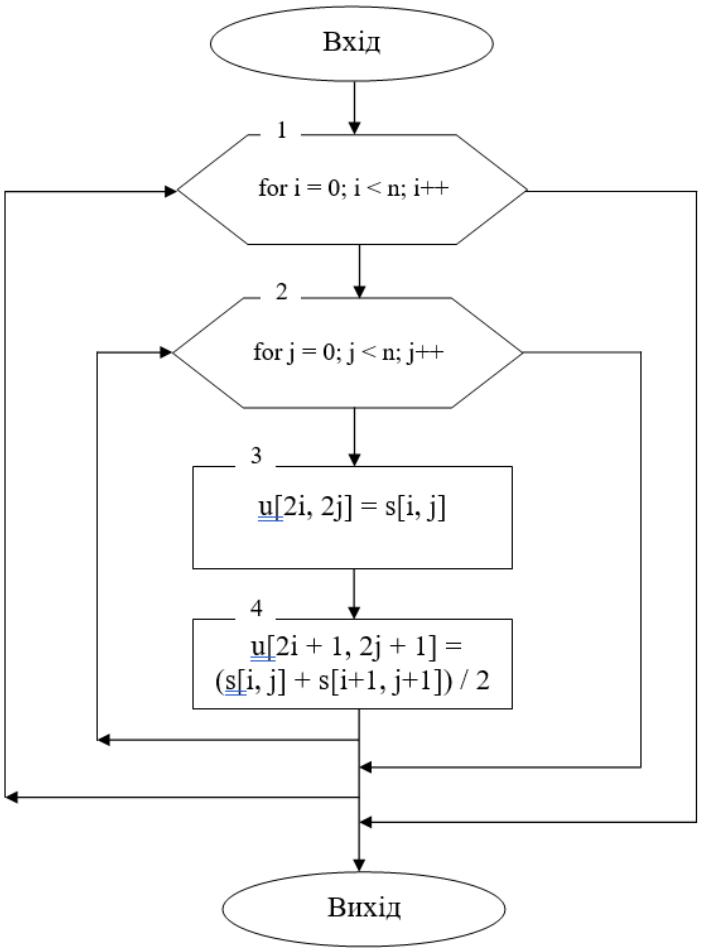


Рисунок 2.18 – Алгоритм декодування блоку S` закодованого з частковим зберіганням компонент U та V для блоку U`

При декодуванні також потрібно враховувати, що ширина або висота блоку може бути непарним числом.

2.7 Висновки

У другому розділі було проведено аналіз інформаційного забезпечення програми для ущільнення зображень з використанням рекурсивного методу. Було розроблено загальну модель роботи програмного продукту за допомогою UML-діаграми діяльності, що відображає послідовність основних дій користувача та системи під час обробки зображень. Крім того, був сформульований основний алгоритм роботи застосунку, який описує логіку взаємодії користувача з інтерфейсом і процес ущільнення зображень.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програми

У даний час найбільш поширеними мовами програмування для ПК є Java та C#.

Java – об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Мова значно запозичила синтаксис із C і C++ [25]. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, покладено на віртуальну машину. Java, передусім, розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. В разі необхідності таких дій, java дозволяє викликати підпрограми, написані на інших мовах програмування.

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Хейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft) [26]. Синтаксис C#, так як і синтаксис Java, близький до C++ . Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML.

Вище описані мови програмування дуже подібні і мають багато спільного. Але також є деякі відмінності. C# має такі переваги над Java :

− Більша орієнтованість під ОС Windows. Це є досить значною перевагою у нашому випадку, оскільки гра розробляється саме для даної операційної системи.

− Делегати. У Java є їх аналог – функції зворотного виклику, але делегати значно зручніші і надійніші.

− Властивості. Вони дозволяють керувати читанням та записом поля класу. Завдяки їм можна зробити поле доступним тільки для читання або запису, задати певну умову присвоєння значення полю класу.

− Оператори as та is. Оператор as дозволяє приводити один тип до іншого. На відміну від звичайного приведення типів, яке присутнє у C# та Java, коли тип, до якого перетворюється записується перед посиланням в дужках, він не генерує виключення, коли неможливо здійснити перетворення типів. Замість цього він повертає значення null. Оператор is дозволяє визначити, чи можна привести один тип до іншого.

Visual Studio — це флагманське середовище розробки для C# і платформи .NET. Воно є офіційним продуктом Microsoft і надає найповніший функціонал для створення як простих, так і масштабних застосунків [27].

Ключові переваги:

- Повна інтеграція з .NET Framework / .NET Core / .NET 5+ - дозволяє розробляти будь-які типи застосунків — консольні, десктопні (Windows Forms, WPF), веб-застосунки, служби;
- Візуальний дизайнер - має зручний drag-and-drop дизайнер форм для Windows Forms або XAML-дизайнер для WPF, що прискорює розробку GUI;
- Можливості налагодження (debugging) - дає змогу ставити точки зупину, переглядати значення змінних у реальному часі, аналізувати стек викликів, обробку винятків тощо;
- Інструменти для аналізу продуктивності - дозволяють оцінити швидкодію алгоритмів ущільнення зображень, знайти вузькі місця;
- Розширення та інтеграція - підтримка NuGet, Git, Azure DevOps, Docker та ін. Можна підключити бібліотеки для обробки зображень (наприклад, System.Drawing, ImageSharp);
- Інтегроване тестування - підтримка MSTest, NUnit, xUnit — можна писати юніт-тести для перевірки алгоритмів рекурсивного ущільнення;
- Велика спільнота та документація - будь-яке питання вже, швидше за все, має відповідь на StackOverflow або в офіційній документації.

VS Code — це легкий редактор коду від Microsoft, орієнтований на універсальність і гнучкість [28]. За замовчуванням не підтримує C#, але це легко виправляється через встановлення розширення C# (OmniSharp).

Переваги:

– Легка вага та швидкість - ідеально для слабших машин або простих проєктів;

– Розширення - понад 30 000 плагінів: підтримка Git, автодоповнення, AI-кодування (Copilot), форматування коду тощо;

– Вбудований термінал - зручно запускати команди .NET CLI, компіляцію, публікацію, тестування;

– Кросплатформеність працює на Windows, Linux, macOS;

– Гарна підтримка Markdown, JSON, XML, що корисно для документації або конфігурацій.

Недоліки

– Немає вбудованого GUI-дизайнера;

– Більш складне налаштування для новачків;

– Обмежені можливості налагодження GUI-застосунків.

Rider — це кросплатформене IDE від JetBrains (творців IntelliJ IDEA, PyCharm), побудоване на базі ReSharper [29].

Переваги:

– Потужні інструменти рефакторингу та аналізу коду**;**

– Інтелектуальні підказки**:** Rider постійно пропонує оптимізації коду, виявляє помилки ще до компіляції;

– Кросплатформеність**:** працює на Windows, macOS і Linux;

– Вбудована підтримка юніт-тестування, профілювання, баз даних**.**

Недоліки:

- Платне (є безкоштовна версія для студентів);

- Дещо складніше налаштування при роботі з дизайнером форм Windows**;**

- Менш активна спільнота для C#, ніж у Visual Studio.

MonoDevelop це середовище орієнтоване на кросплатформену розробку під

Linux та мобільні пристрої. Воно базується на Mono — відкритій реалізації .NET Framework.

Переваги:

– Розробка під Linux;

– Інтеграція з Xamarin для створення мобільних застосунків на C#.

Недоліки:

– Слаба підтримка Windows Forms/WPF;

– Неактуальне для десктопних застосунків, особливо в контексті роботи із зображеннями та складними UI.

Вище наведені переваги C# над Java є дуже важливими для розробки застосунку, відповідно до технічного завдання. Отже, для створення розроблюваної програми ущільнення зображень на основі Wavelet-перетворень краще вибрати мову програмування C# та середовище програмування Microsoft Visual Studio.

3.2 Розробка модулів програми

Для того, щоб програма могла виконувати усі поставлені завдання, відповідно до принципів об’єктно-орієнтованого програмування та особливостей мови C# необхідно створити такі класи:

– MainForm;

– WaveletEncoding;

– WaveletDecoding.

Створення цих класів виконується за допомогою Microsoft Visual Studio 2010. На рисунку 3.1 розроблені класи програми.

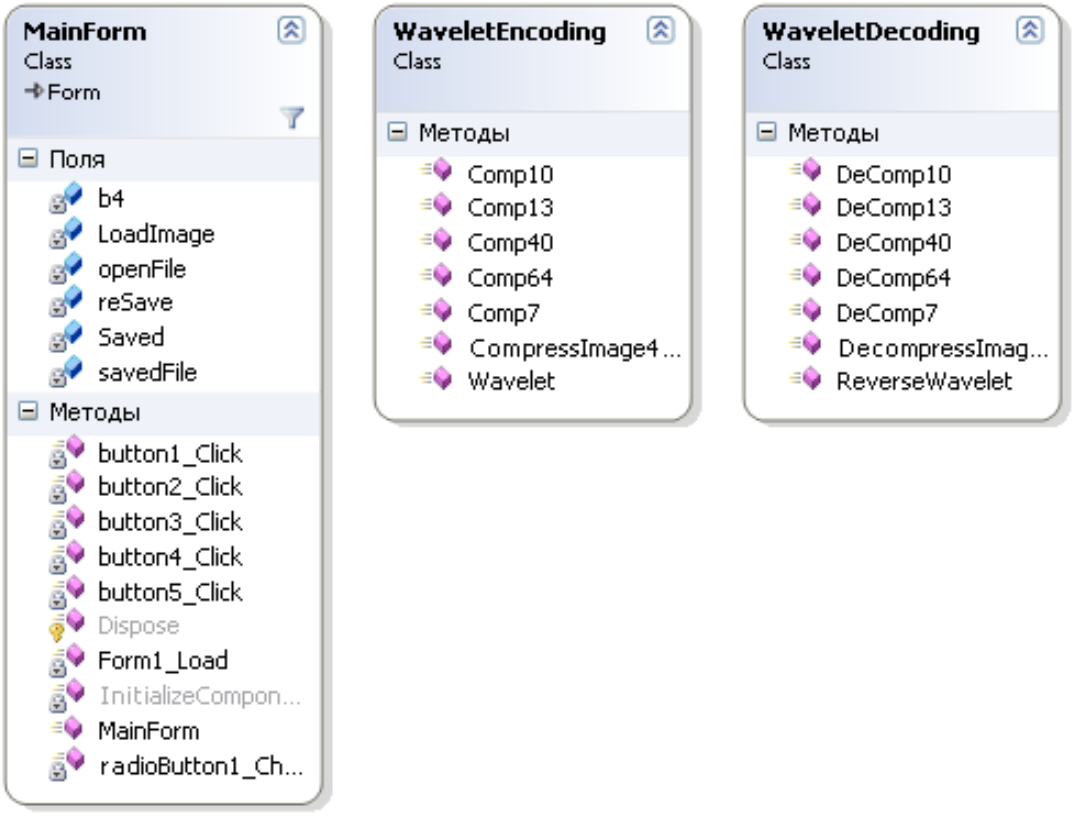


Рисунок 3.1 – Класи програми

Клас MainForm містить поля: b4 – змінна типу bool, яка показує, чи потрібно при ущільненні зберігати компоненту b4; loadImage – змінна типу Bitmap, яка вміщує завантажене для ущільнення або відновлення зображення; openFile – змінна типу string, яка містить шлях до файлу з відкритим зображенням; saveFile – змінна типу string, яка містить шлях до файлу, у який записується відновлене або закодоване зображення; resave – змінна типу bool, яка визначає чи виконувати кодування з можливістю повторного ущільнення відновленого зображення без втрат яксоті.

Клас WaveletEncoding містить методи: Wavelet – виконує перетворення для блоку пікселів 2x2; CompresImage4 – ущільнює зображення з розкладом на 4 компоненти; Comp7 – ущільнює зображення з розкладом на 7 компонент; Comp7 – ущільнює зображення з розкладом на 7 компонент; Comp10 – ущільнює зображення з розкладом на 10 компонент; Comp13 – ущільнює зображення з розкладом на 13 компонент; Comp40 – ущільнює зображення з розкладом на 40

компонент; Comp64 – ущільнює зображення з розкладом на 64 компоненти.

Клас WaveletDecoding містить методи: ReverseWavelet – виконує зворотнє перетворення для блоку пікселів 2x2; DeCompresImage4 – відновлює зображення, розкладене на 4 компоненти; DeComp7 – відновлює зображення, розкладене на 7 компонент; DeComp10 – відновлює зображення, розкладене на 10 компонент; DeComp13 – відновлює зображення, розкладене на 13 компонент; DeComp40 – відновлює зображення, розкладене на 40 компонент; DeComp64 – відновлює зображення, розкладене на 64 компоненти.

Безумовно, головним методом для виконання ущільнення зображення є метод CompresImage4(), який виконує ущільнення блоку пікселів на зображенні. Завдяки цьому його можна використовувати при розробці методів ущільнення з розкладом на іншу кількість компонент. Вхідними даними для методу CompresImage4() є: зображення, яке кодується (ProcessingImage); координати (Xmin,Ymin) та (Xmax, Ymax), які вказують координати прямокутного блока пікселів на зображенні, який кодується; параметр B4, який показує, чи потрібно при ущільненні зберігати компоненту B4; параметр speed, який показує, чи потрібно буде повторно ущільнювати після відновлення зображення, що кодується.

При виконанні відновлення головним є метод DeCompresImage4(). Він виконує відновлення на зображенні закодованого блоку пікселів. Відповідно до пірамідальної схеми розкладу, його використовують методи, які виконують відновлення зображення, розкладеного на будь-яку кількість компонент. Вхідними даними для методу DeCompresImage4() є: зображення, яке кодується (ProcessingImage); координати (Xmin,Ymin) та (Xmax, Ymax), які вказують координати прямокутного блока пікселів на зображенні, який відновлюється.

Лістинг програми наведено у додатку Б.

3.3 Висновки

У третьому розділі було обґрунтовано вибір технологій та інструментів, що використовувалися для розробки програмних модулів системи ущільнення

зображень з використанням рекурсивного методу.

На основі аналізу вимог до функціональності, надійності та зручності використання було обрано мову програмування C# як основну платформу для реалізації програмного забезпечення. Для створення графічного інтерфейсу користувача було використано Windows Forms, що забезпечує простоту у проєктуванні, адаптивність та інтеграцію з іншими модулями.

У розділі також було описано реалізацію основних програмних модулів, зокрема:

– модуль для рекурсивного ущільнення зображень, який виконує розбиття зображення на блоки, оцінювання подібності та застосування алгоритму ущільнення до кожного блоку;

Таким чином, обрані інструменти та реалізовані модулі дозволили створити ефективне, структуроване і зручне програмне рішення, яке забезпечує автоматизоване стиснення зображень із можливістю відстеження всіх етапів обробки.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування

Тестування програмного забезпечення є завершальним етапом життєвого циклу розробки, що має вирішальне значення для оцінки його якості, стабільності та відповідності технічним вимогам. Вибір методів тестування залежить від архітектури системи, типу вхідних і вихідних даних, а також особливостей взаємодії користувача з програмою.

Оскільки розроблений застосунок виконує ущільнення зображень за допомогою рекурсивного методу, основна увага приділяється перевірці коректності алгоритмічної обробки зображень, збереження структури, якості після стиснення, а також відповідності отриманого результату очікуваному. Тому пріоритетним є функціональне тестування, яке дозволяє впевнитися, що алгоритм рекурсивного ущільнення виконується правильно, параметри обробки застосовуються вірно, а результат зберігається у заданому форматі.

Тестування на відмови є важливим для перевірки стабільності роботи у випадку помилок, таких як некоректне зображення, нестача ресурсів або спроба обробки файлів з обмеженим доступом. Воно дозволяє переконатися, що застосунок здатен адекватно обробляти виняткові ситуації, не завершуючи роботу аварійно.

Швидкодія також є критичним показником, особливо при обробці великих зображень або при виконанні рекурсивних обчислень. Тестування швидкодії дозволяє виявити вузькі місця в продуктивності та оптимізувати код.

Оскільки застосунок має графічний інтерфейс, необхідне тестування UI, яке включає перевірку доступності всіх елементів керування, їх логічного розташування, адаптивності до різних розмірів вікна та правильності відображення результатів ущільнення.

Додатково важливо провести тестування сумісності, що дозволяє впевнитися у працездатності застосунку на різних версіях ОС Windows та з різними конфігураціями обладнання.

Отже, комплексне застосування функціонального тестування, тестування на відмову, швидкодії, інтерфейсу та сумісності забезпечує повну перевірку програмного продукту, що працює за принципом рекурсивного ущільнення зображень, і гарантує його якість та зручність для кінцевого користувача.

4.2 Тестування програмного застосунку

Для проведення тестування було встановлено фінальну версію програмного застосунку для ущільнення зображень з використанням рекурсивного методу на комп’ютер із операційною системою Windows 11.

На першому етапі було проведено функціональне тестування, під час якого перевірялися основні можливості програми. Зокрема, успішно пройдено перевірку відкриття зображення, коректного запуску алгоритму рекурсивного ущільнення, збереження отриманого результату у вибраний формат, а також відображення відповідних повідомлень про успішне завершення або помилки при обробці.

Другим етапом було тестування відмов, у рамках якого моделювалися ситуації, що можуть порушити стабільну роботу програми. Серед них — спроба обробки пошкодженого файлу зображення, відсутність доступу до каталогу збереження, а також обрив процесу ущільнення. Усі виняткові ситуації були належним чином оброблені: користувач отримував повідомлення про помилку, а додаток продовжував роботу без критичних збоїв.

На третьому етапі здійснено тестування швидкодії. Було зафіксовано, що процес ущільнення зображення середньої роздільної здатності виконувався в межах 0.3–0.6 секунд, що свідчить про високу продуктивність реалізованого алгоритму. Затримок при роботі з інтерфейсом не спостерігалося навіть на комп’ютерах із середніми технічними характеристиками.

Також проведено тестування графічного інтерфейсу користувача. Було перевірено правильність відображення всіх елементів, адаптивність інтерфейсу до різних розмірів вікна, стабільність перемикання між режимами та коректність візуального відображення результатів ущільнення.

Дослідимо залежність коефіцієнту ущільнення від кількості компонент

перетворення, а саме:

- розклад зображення на 4 компоненти;
- розклад зображення на 10 компонент;

Результати наведено в табл. 4.1, приклади зображень в додатку Д.

Таблиця 4.1 – Тестування програми на прикладі файлу Lena

Файл та тип ущільнення	Тип зображення	Початковий розмір даних зображення, байт	Розмір після ущільнення, байт	Кількість компонент	Погіршення візуальної якості
Lena512.bmp	напівтонове	786572	177650	4	Не помітно
Lena512.bmp	напівтонове	786572	148519	10	Не помітно
Lena512.bmp (Jpeg)	напівтонове	786572	82912	-	Не помітно
lena512color.tiff	кольорове	786572	390990	4	Не помітно
lena512color.tiff (WH8)	кольорове	786572	296452	10	Не помітно
lena512color.tiff (Jpeg)	кольорове	786572	95646	-	Не помітно

Аналіз результатів показує, що об’єм зображення зменшується в 1,5-2 рази, що відповідає технічному завданню на роботу.

Збіг теоретичних і практичних результатів роботи програми свідчить про її повну працездатність.

Таким чином, результати проведеного тестування підтверджують відповідність програмного застосунку функціональним вимогам, його стабільність, ефективність і зручність у використанні.

4.3 Розробка інструкції для користувача програмного забезпечення

Дана інструкція призначена для користування програмним продуктом «Wavelet» – застосунок для ущільнення зображень з використанням рекурсивного методу. Для інсталяції застосунка необхідно виконати такі кроки:

1. Завантажте програмний застосунок.

2. Запустіть програму, після чого з’явиться головне вікно програмного застосунку (див. рисунок 4.1).

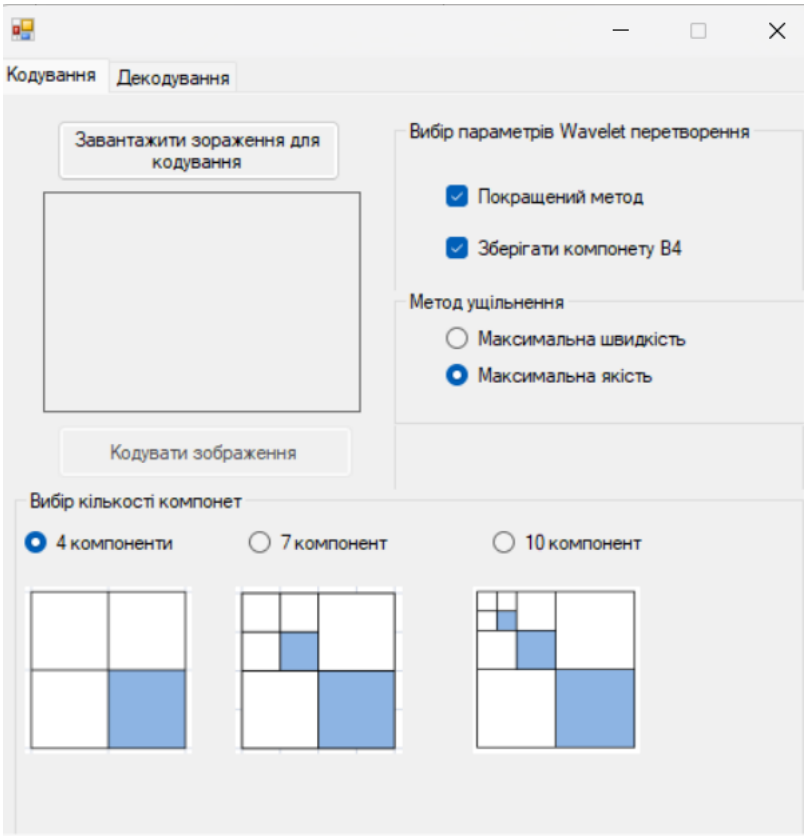


Рисунок 4.1 – Головне вікно

Основними можливостями розробленого застосунку є ущільнення зображень із використанням рекурсивного методу, збереження резервної копії оригінального зображення перед виконанням операцій.

Використання функцій додатку можливе за допомогою пунктів меню, дані елементи інтерфейсу наведено на рисунку 4.2.

Меню складається з 2 пунктів:

– «Кодування»;

– «Декодування».

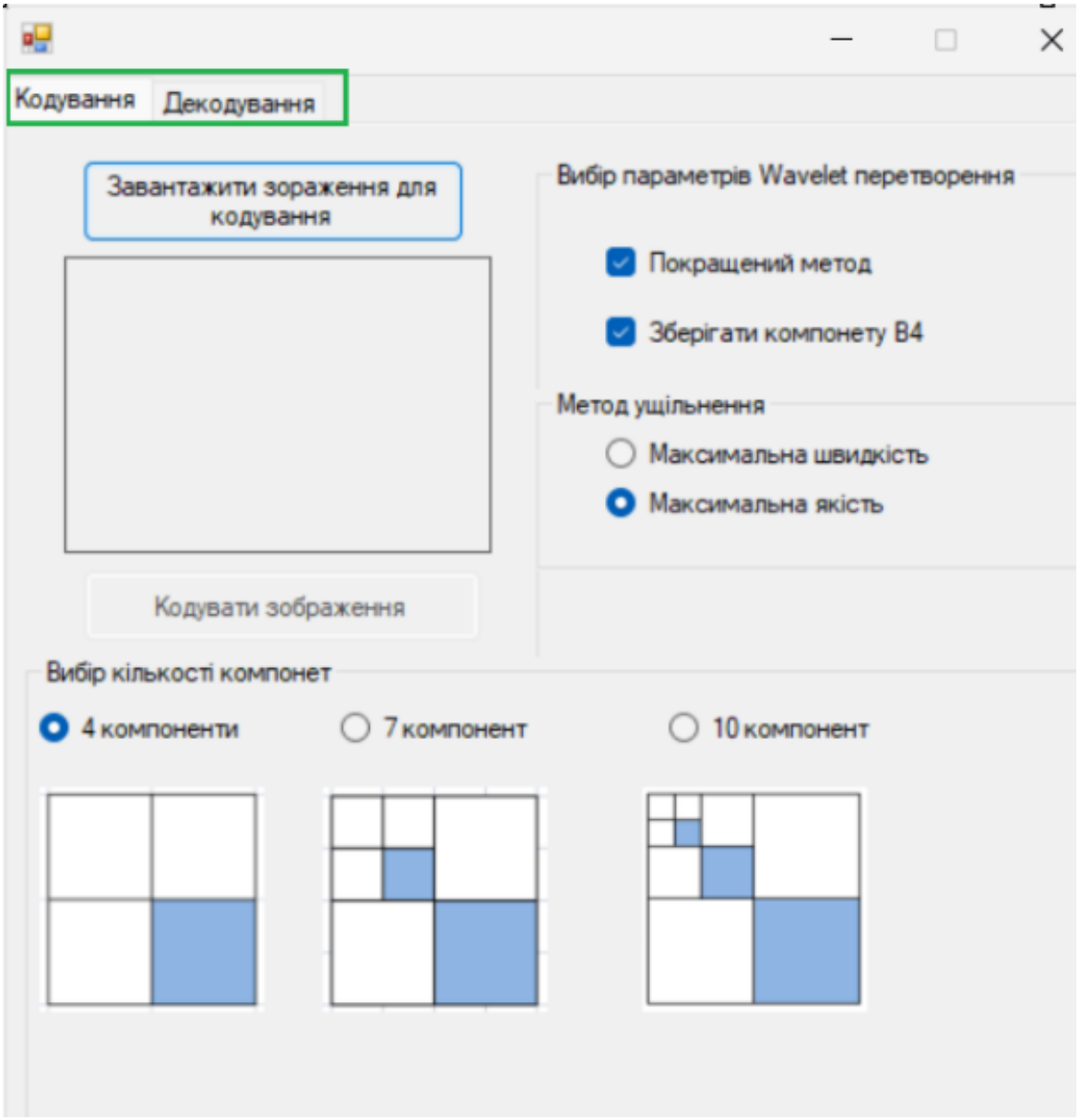


Рисунок 4.2 – Стрічка меню

На рисунку 4.3 продемонстровано меню «Декодування».

Рисунок 4.3 – Пункт меню «Декодування»

Кнопка для завантаження зображення (див. рисунок 4.4).

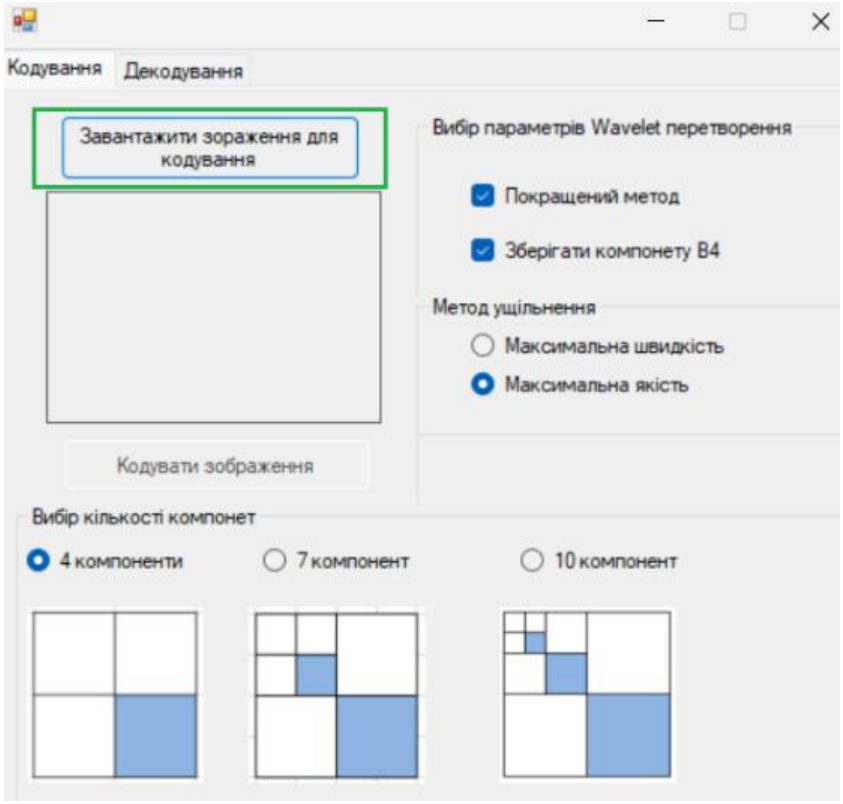


Рисунок 4.4 – Кнопка для завантаження зображення

На рисунку 4.5 продемонстровано параметри «Wavelet» перетворень.

Є такий вибір пеаметрів як, покращений метод та збереження компоненти.

Також такі методи ущільнення як максимальна швидкість та максимальна якість.

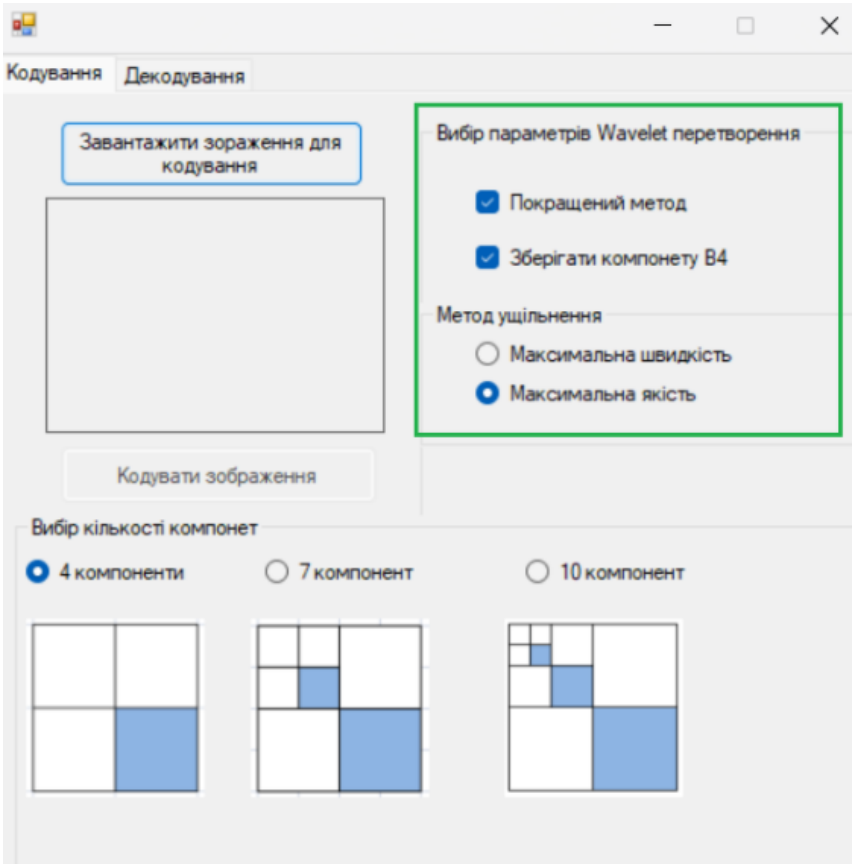


Рисунок 4.5 – Параметри «Wavelet» перетворень

На рисунку 4.6 продемонстровано вибір кількості компонент.

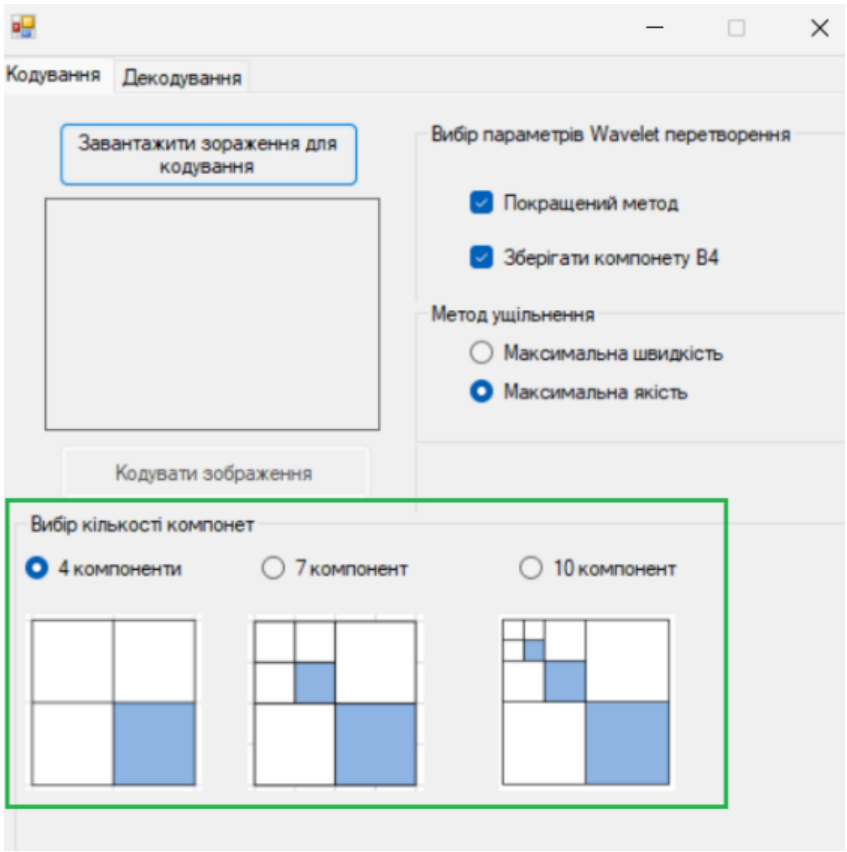


Рисунок 4.6 – Вибір кількості компонент

Інтерфейс застосунку орієнтований на максимальну простоту та інтуїтивність, що дозволяє користувачам ефективно працювати з операціями ущільнення зображень навіть без спеціальних технічних знань.

Таким чином, дана інструкція охоплює основні аспекти використання програми: запуск, завантаження зображення, вибір параметрів ущільнення, перегляд результату, збереження зміненого файлу, створення резервної копії оригіналу, зміна теми оформлення та перегляд журналу операцій. Застосунок спроєктовано таким чином, щоб забезпечити користувачу зручне та безпечне ущільнення зображень із застосуванням рекурсивного методу без необхідності ручного втручання в алгоритм обробки.

4.4 Висновки

У четвертому розділі було проведено тестування програмних модулів системи ущільнення зображень. Було здійснено функціональне тестування, перевірку поведінки застосунку в умовах відмов, тестування швидкодії, оцінку зручності графічного інтерфейсу користувача та тестування сумісності з різними версіями операційної системи. Особливу увагу приділено реакції програми на типові та виняткові ситуації, пов’язані з обробкою графічних файлів. Результати тестування підтвердили правильність реалізації основного функціоналу, стабільність роботи та високу продуктивність системи ущільнення. Також було підготовлено детальну інструкцію користувача для забезпечення простоти й ефективності взаємодії із програмою.

ВИСНОВКИ

У рамках бакалаврської кваліфікаційної роботи було розроблено програмний застосунок, призначений для ущільнення зображень із використанням рекурсивного методу. Для реалізації було обрано мову програмування C# та середовище розробки Microsoft Visual Studio, яке забезпечило високу зручність у написанні, тестуванні та налагодженні коду, а також ефективну роботу з графічним інтерфейсом користувача на платформі Windows.

Реалізація бакалаврської кваліфікаційної роботи виконана відповідно до методичних вказівок. У процесі роботи було проведено аналіз існуючих аналогічних програм, виявлено їхні обмеження та недоліки, на основі чого сформульовано вимоги до функціоналу власної розробки.

Було обґрунтовано вибір мови C# завдяки її потужним можливостям у роботі з графікою, підтримці об’єктно-орієнтованого підходу, а також можливості інтеграції із системними компонентами Windows. Microsoft Visual Studio було обрано як основне середовище розробки завдяки потужному інструментарію, зручному інтерфейсу, інтегрованому відлагодженню та широкій підтримці бібліотек.

У межах бакалаврської кваліфікаційної роботи було виконано:

- розроблено окремі програмні модулі для виконання рекурсивного стиснення зображень;

- проведено повне тестування програмного продукту відповідно до вимог технічного завдання.

Результати тестування підтвердили працездатність застосунку, стабільну роботу, високу швидкодію, точність реалізації алгоритмів ущільнення, а також зручність взаємодії з інтерфейсом. Крім того, було підготовлено детальну інструкцію користувача, що дозволяє легко освоїти функціонал навіть без спеціальної підготовки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sayood K. Introduction to Data Compression. – 5th ed. – Morgan Kaufmann, 2017. – 790 p.

2. Ліщинська Л. Б., Слободян Д. І. Розробка програмного застосунку для ущільнення зображень з використанням рекурсивного методу // Молодь в науці: дослідження, проблеми, перспективи (МН-2025) [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/24807

3. Gonzalez R. C., Woods R. E. Digital Image Processing. – 4th ed. – Pearson, 2018. – 1024 p.

4. Image Compression – Tutorialspoint [Електронний ресурс]. URL: https://www.tutorialspoint.com/digital_image_processing/image_compression.htm

5. Digital Image Processing – Resources and Examples [Електронний ресурс]. URL: https://www.imageprocessingplace.com/DIP-4E/dip4e_main_page.htm

6. Peak signal-to-noise ratio – Wikipedia [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

7. Information technology – Digital compression and coding of continuous-tone still images – Requirements and guidelines. ITU-T T.81 [Електронний ресурс]. – URL: https://www.w3.org/Graphics/JPEG/itu-t81.pdf

8. RIOT – Radical Image Optimization Tool [Електронний ресурс]. URL: https://riot-optimizer.com/

9. FileOptimizer – Nikkhokkho Projects [Електронний ресурс]. URL: https://nikkhokkho.sourceforge.io/static.php?page=FileOptimizer

10. Caesium Image Compressor – SaeraSoft [Електронний ресурс]. URL: https://saerasoft.com/caesium/

11. ImageOptim. [Електронний ресурс]. URL: https://imageoptim.com/

12. Photoshop User Guide – Adobe [Електронний ресурс]. URL: https://helpx.adobe.com/photoshop/user-guide.html

13. Nelson M., Gailly J.-L. The Data Compression Book. – 2nd ed. – M&T Books, 1995. – 576 p.

14. Huffman Coding – GeeksforGeeks [Електронний ресурс]. URL: https://www.geeksforgeeks.org/huffman-coding/

15. Run Length Encoding – GeeksforGeeks [Електронний ресурс]. URL: https://www.geeksforgeeks.org/run-length-encoding/

16. LZW (Lempel–Ziv–Welch) Compression – GeeksforGeeks [Електронний ресурс]. URL: https://www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/

17. Image Compression – Tutorialspoint [Електронний ресурс]. URL: https://www.tutorialspoint.com/digital_image_processing/image_compression.htm

18. Sayood K. Introduction to Data Compression. – 5th ed. – Morgan Kaufmann, 2017. – 790 p.

19. ITU-T T.81 – JPEG Standard [Електронний ресурс]. URL: https://www.w3.org/Graphics/JPEG/itu-t81.pdf

20. JPEG Official Website [Електронний ресурс]. URL: https://www.jpeg.org/jpeg/index.html

21. Wavelet Tutorial – Wavelet.org [Електронний ресурс]. URL: https://www.wavelet.org/tutorial/

22. YUV – Wikipedia [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/YUV

23. Continuous Wavelet Transform – MathWorks [Електронний ресурс]. – URL: https://www.mathworks.com/help/wavelet/continuous-wavelet-transform.html

24. Haar Wavelet – Wikipedia [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Haar_wavelet

25. Niemeyer P., Leuck D. Learning Java. – 5th ed. – O’Reilly Media, 2020. – 520 p.

26. Skeet J. C# in Depth. – 5th ed. – Manning Publications, 2022. – 632 p.

27. Visual Studio Documentation – Microsoft [Електронний ресурс]. URL: https://docs.microsoft.com/en-us/visualstudio/

28. Visual Studio Code Documentation [Електронний ресурс]. – URL: https://code.visualstudio.com/docs

29. JetBrains Rider Documentation [Електронний ресурс]. URL: https://www.jetbrains.com/rider/documentation/

ДОДАТКИ

62

ДОДАТОК А (обов'язковий). Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
« 21 » березня 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Розробка програмного застосунку для ущільнення зображень з використанням рекурсивного методу»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. Майданюк В.П.
« 21 » березня 2025 року.
Виконав:
студент гр. 1ПІ-21б Слободян Д.І.
« 21 » березня 2025 року.

м. Вінниця – 2025 рік

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка програмного забезпечення для дослідження алгоритму ущільнення зображень з використанням ркурсивного методу».

Галузь застосування – розробка та дослідження алгоритмів ущільнення зображень з використанням рекурсивних методів для оптимізації обробки графічних даних у програмному забезпеченні.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 р. ректора ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою бакалаврської кваліфікаційної роботи є підвищення коефіцієнту ущільнення зображень з використанням рекурсивного перетворення.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР:

1. Sayood K. Introduction to Data Compression. – 5th ed. – Morgan Kaufmann, 2017. – 790 p.

2. Ліщинська Л. Б., Слободян Д. І. Розробка програмного застосунку для ущільнення зображень з використанням рекурсивного методу // Молодь в науці: дослідження, проблеми, перспективи (МН-2025) [Електронний ресурс]. URL : https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/24807.

3. Gonzalez R. C., Woods R. E. Digital Image Processing. – 4th ed. – Pearson, 2018. – 1024 p.

4. Image Compression – Tutorialspoint [Електронний ресурс]. URL: https://www.tutorialspoint.com/digital_image_processing/image_compression.htm

5. Digital Image Processing – Resources and Examples [Електронний ресурс]. URL: https://www.imageprocessingplace.com/DIP-4E/dip4e_main_page.htm

5. Технічні вимоги

– операційна система – Windows;

– середовище розробки – Visual Studio ;

– мова програмування – C#;

– коефіцієнт ущільнення - 2.

6. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

–пояснювальна записка до бакалаврської кваліфікаційної роботи;
–технічне завдання;
–лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку програмних застосунків для ущільнення зображень	25.03.25 – 31.03.25
2	Розробка методів, моделі та алгоритму роботи програмного забезпечення	03.04.25 – 14.04.25
3	Варіантний аналіз та вибір мови програмування розробки	17.04.25 – 21.04.25
4	Розробка програмного забезпечення	24.04.25 – 12.05.25
5	Тестування програмного забезпечення	15.05.25 – 19.05.25
6	Оформлення матеріалів до захисту БКР	22.05.25 – 30.05.25

10. Порядок контролю та прийняття.

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

65

ДОДАТОК Б.

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Розробка програмного застосунку для ущільнення зображень з використанням рекурсивного методу»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, факультет інформаційних технологій та комп'ютерної інженерії, 1ПП-21Б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 13 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Майданюк В.П. к.т.н., доц. каф. ПЗ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Слободян Д.І.
(підпис) (прізвище, ініціали)

ДОДАТОК В (довідниковий). Лістинг програми

```
using System;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace Wavelet
{
   public partial class MainForm : Form
   {
      string openFile = "";
      string savedFile = "";
      Bitmap LoadImage;
      Bitmap Saved;
      bool b4 = true; //Зберігання компонент B4
      bool reSave = true;

      public MainForm()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         //textBox1.Clear();
         OpenFileDialog openFileDialog1 = new OpenFileDialog();
//завантаження незакодованого зображення
         openFileDialog1.Filter = "bmp files (*.bmp)|*.bmp";
```

```
            openFileDialog1.InitialDirectory = Application.StartupPath;
            openFileDialog1.ShowDialog();
            openFile = openFileDialog1.FileName;
            if (openFile != "")
            {
               LoadImage = new Bitmap(openFile);
               //for (int i = 0; i < LoadImage.Height; i++)
               //{
               //    for (int j = 0; j < LoadImage.Width; j++)
               //    {
               //        Color c = LoadImage.GetPixel(j, i);
               //        LoadImage.SetPixel(
               //            j,
               //            i,
               //            Color.FromArgb(
               //                (int)(0.299 * c.R + 0.587 * c.G + 0.114 * c.B),
               //                (int)(-0.14713 * c.R - 0.28886 * c.G + 0.436 * c.B + 128),
               //                (int)(0.615 * c.R - 0.51499 * c.G - 0.10001 * c.B + 128)
               //            )
               //        );
               //    }
               //}
               pictureBox1.Image = LoadImage;
               button3.Enabled = true;
               System.IO.File.Copy(openFile, openFile.Split('.')[0] + "++.bmp", true);
               FileStream fs = new FileStream(openFile.Split('.')[0] + "++.bmp",
FileMode.Open);
               LoadImage.Save(fs, System.Drawing.Imaging.ImageFormat.Bmp);
            }
            else
```

```
            pictureBox1.Image = null;
        }

        private void button2_Click(object sender, EventArgs e)  //завантаження зображення, яке потрібно відновити
        {
            OpenFileDialog openFileDialog1 = new OpenFileDialog();
            openFileDialog1.Filter = "bmp files (*.bmp)|*.bmp";
            openFileDialog1.InitialDirectory = Application.StartupPath;
            openFileDialog1.ShowDialog();
            openFile = openFileDialog1.FileName;
            if (openFile != "")
            {
                LoadImage = new Bitmap(openFile);
                pictureBox2.Image = LoadImage;
                button4.Enabled = true;
            }

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)//кодування зображення
        {
            b4 = checkBox1.Checked;
            SaveFileDialog saveFileDialog1 = new SaveFileDialog(); //завантаження
```

```
зображення
            saveFileDialog1.Filter = "bmp files (*.bmp)|*.bmp";
            saveFileDialog1.Filter = "bmp files (*.bmp)|*.bmp";
            saveFileDialog1.InitialDirectory = Application.StartupPath;
            saveFileDialog1.ShowDialog();
            savedFile = saveFileDialog1.FileName;
            if (savedFile != "")
            {
               //вибір кількості компонент для кодування
               if (r4.Checked)
               {
                  System.IO.File.Copy(openFile, savedFile.Split('.')[0] + ".4." +
(useYuv_checkbox.Checked ? "yuv." : "") + "bmp", true);
                  FileStream fs = new FileStream(savedFile.Split('.')[0] + ".4." +
(useYuv_checkbox.Checked ? "yuv." : "") + "bmp", FileMode.Open);
                  Saved = new Bitmap(fs);

                  if(useYuv_checkbox.Checked)
                     convertToYuv(ref Saved);

                  WaveletEncoding.CompressImage4Comp(0, 0, Saved.Width,
Saved.Height, ref Saved, b4, reSave);

                  Saved.Save(fs, System.Drawing.Imaging.ImageFormat.Bmp);
                  fs.Close();
               }
               if (r7.Checked)
               {
                  System.IO.File.Copy(openFile, savedFile.Split('.')[0] + ".7." +
(useYuv_checkbox.Checked ? "yuv." : "") + "bmp", true);
```

```
                FileStream fs = new FileStream(savedFile.Split('.')[0] + ".7.bmp" +
(useYuv_checkbox.Checked ? "yuv." : "") + "", FileMode.Open);
                Saved = new Bitmap(fs);

                if (useYuv_checkbox.Checked)
                    convertToYuv(ref Saved);

                WaveletEncoding.Comp7(0, 0, Saved.Width, Saved.Height, ref
Saved, b4, reSave);

                Saved.Save(fs, System.Drawing.Imaging.ImageFormat.Bmp);
                fs.Close();
            }
            if (r10.Checked)
            {
                System.IO.File.Copy(openFile, savedFile.Split('.')[0] + ".10." +
(useYuv_checkbox.Checked ? "yuv." : "") + "bmp", true);
                FileStream fs = new FileStream(savedFile.Split('.')[0] + ".10." +
(useYuv_checkbox.Checked ? "yuv." : "") + "bmp", FileMode.Open);
                Saved = new Bitmap(fs);

                if (useYuv_checkbox.Checked)
                    convertToYuv(ref Saved);

                WaveletEncoding.Comp10(0, 0, Saved.Width, Saved.Height, ref
Saved, b4, reSave);

                Saved.Save(fs, System.Drawing.Imaging.ImageFormat.Bmp);
                fs.Close();
            }
```

```
        }
    }

    private void button4_Click(object sender, EventArgs e) //відновлення закодованого зображення
    {
        SaveFileDialog saveFileDialog1 = new SaveFileDialog(); //завантаження зображення
        saveFileDialog1.Filter = "bmp files (*.bmp)|*.bmp";
        saveFileDialog1.Filter = "bmp files (*.bmp)|*.bmp";
        saveFileDialog1.InitialDirectory = Application.StartupPath;
        saveFileDialog1.ShowDialog();
        savedFile = saveFileDialog1.FileName;
        if (savedFile != "")
        {

            System.IO.File.Copy(openFile, savedFile, true);
            FileStream fs = new FileStream(savedFile, FileMode.Open);
            Saved = new Bitmap(fs);

            string[] splitedFileNmae = openFile.Split('.');

            switch (splitedFileNmae[1])
            {
                case "4":
                    WaveletDecoding.DecompressImage4Comp(0, 0, Saved.Width, Saved.Height, ref Saved);
                    break;

                case "7":
```

```
                WaveletDecoding.DeComp7(0, 0, Saved.Width, Saved.Height, ref
Saved);
                break;

            case "10":
                WaveletDecoding.DeComp10(0, 0, Saved.Width, Saved.Height,
ref Saved);
                break;



        }

        if (splitedFileNmae[2] == "yuv")
        {
            for (int i = 0; i < Saved.Height; i++)
            {
                for (int j = 0; j < Saved.Width; j++)
                {
                    Color c = Saved.GetPixel(j, i);
                    int r = (int)(c.R + 2.03211 * (c.B - 128));
                    int g = (int)(c.R - 0.39465 * (c.G - 128) - 0.58060 * (c.B - 128));
                    int b = (int)(c.R + 2.03211 * (c.G - 128));

                    Saved.SetPixel(
                        j,
                        i,
                        Color.FromArgb(
                            (r > 255 ? 255 : (r < 0 ? 0 : r)),
                            (g > 255 ? 255 : (g < 0 ? 0 : g)),
                            (b > 255 ? 255 : (b < 0 ? 0 : b))
```

```
                    )
                );
            }
        }
    }

    Saved.Save(fs, System.Drawing.Imaging.ImageFormat.Bmp);
    fs.Close();
    pictureBox9.Image = Saved;
    label_ar.Text = "СКВ = ";
  }
}

private void button5_Click(object sender, EventArgs e)
{
  OpenFileDialog openFileDialog1 = new OpenFileDialog();
//завантаження незакодованого зображення
  openFileDialog1.Filter = "bmp files (*.bmp)|*.bmp";
  openFileDialog1.InitialDirectory = Application.StartupPath;
  openFileDialog1.ShowDialog();
  openFile = openFileDialog1.FileName;
  label_ar.Text = "СКВ = ";
  if (openFile != "")
  {
    Bitmap originImage = new Bitmap(openFile);
    double sum = 0;
    Color restored, origin;
    for (int y = 0; y < Saved.Height; y++)
    {
      for (int x = 0; x < Saved.Width; x++)
```

```
                {
                    restored = Saved.GetPixel(x, y);
                    origin = originImage.GetPixel(x, y);
                    sum += Math.Pow((restored.R - origin.R), 2);
                    sum += Math.Pow((restored.G - origin.G), 2);
                    sum += Math.Pow((restored.B - origin.B), 2);
                }

            }
            label_ar.Text = "CKB = " + (sum / (Saved.Height * Saved.Width *3)).ToString();
        }
    }

    private void radioButton1_CheckedChanged(object sender, EventArgs e)
    {
        reSave = !reSave;
    }


    private void convertToYuv(ref Bitmap bitmap)
    {
        for (int i = 0; i < bitmap.Height; i++)
        {
            for (int j = 0; j < bitmap.Width; j++)
            {
                Color c = bitmap.GetPixel(j, i);
                bitmap.SetPixel(
                    j,
                    i,
```

```
                    Color.FromArgb(
                        (int)(0.299 * c.R + 0.587 * c.G + 0.114 * c.B),
                        (int)(-0.14713 * c.R - 0.28886 * c.G + 0.436 * c.B + 128),
                        (int)(0.615 * c.R - 0.51499 * c.G - 0.10001 * c.B + 128)
                    )
                );
            }
        }
    }

    private void r4_CheckedChanged(object sender, EventArgs e)
    {

    }

    private void pictureBox6_Click(object sender, EventArgs e)
    {

    }
    private void groupBox1_Enter(object sender, EventArgs e)
    {
    }
}
```

ДОДАТОК Г (довідниковий). Шкали квантування різницевих компонент

Інтервал різницевих компонент	Квантоване значення
(-1)-(+1)	0
2-3	2
4-7	5
8-15	12
16-31	24
32-63	48
64-127	96
128-255	192
(-2)-(-3)	-2
(-4)-(-7)	-5
(-8)-(-15)	-12
(-16)-(-31)	-24
(-32)-(-63)	-48
(-64)-(-127)	-96
(-128)-(-255)	-192

ДОДАТОК Д (обов’язковий). Приклади зображень

а)

б)

Рисунок Ж.1 – Кількість компонент 4, коефіцієнт ущільнення 2:

а) початкове зображення; б) відновлене зображення

а)

б)

Рисунок Ж.2 – Кількість компонент 10, коефіцієнт ущільнення 2,6:
а) початкове зображення; б) відновлене зображення

ДОДАТОК Ж (обов’язковий). Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ УЩІЛЬНЕННЯ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ РЕКУРСИВНОГО МЕТОДУ

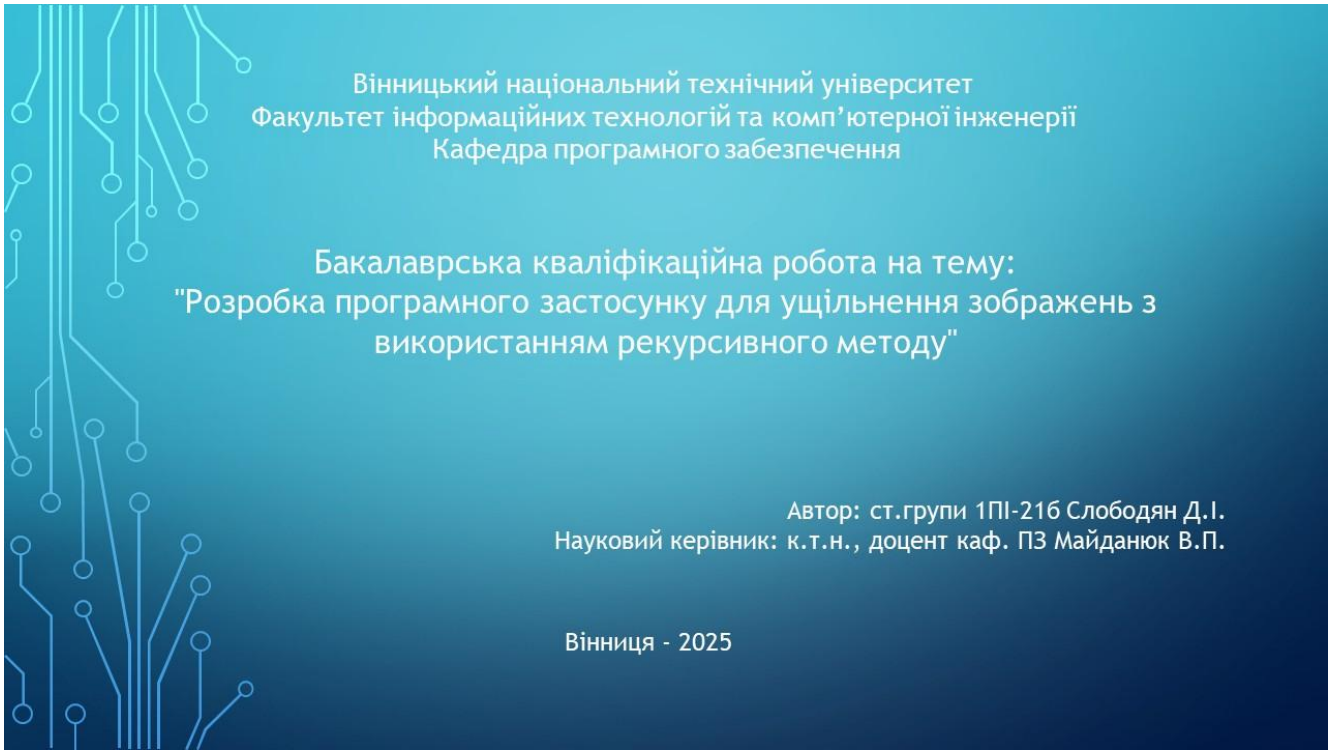


Рисунок Г.1 – Титульний аркуш

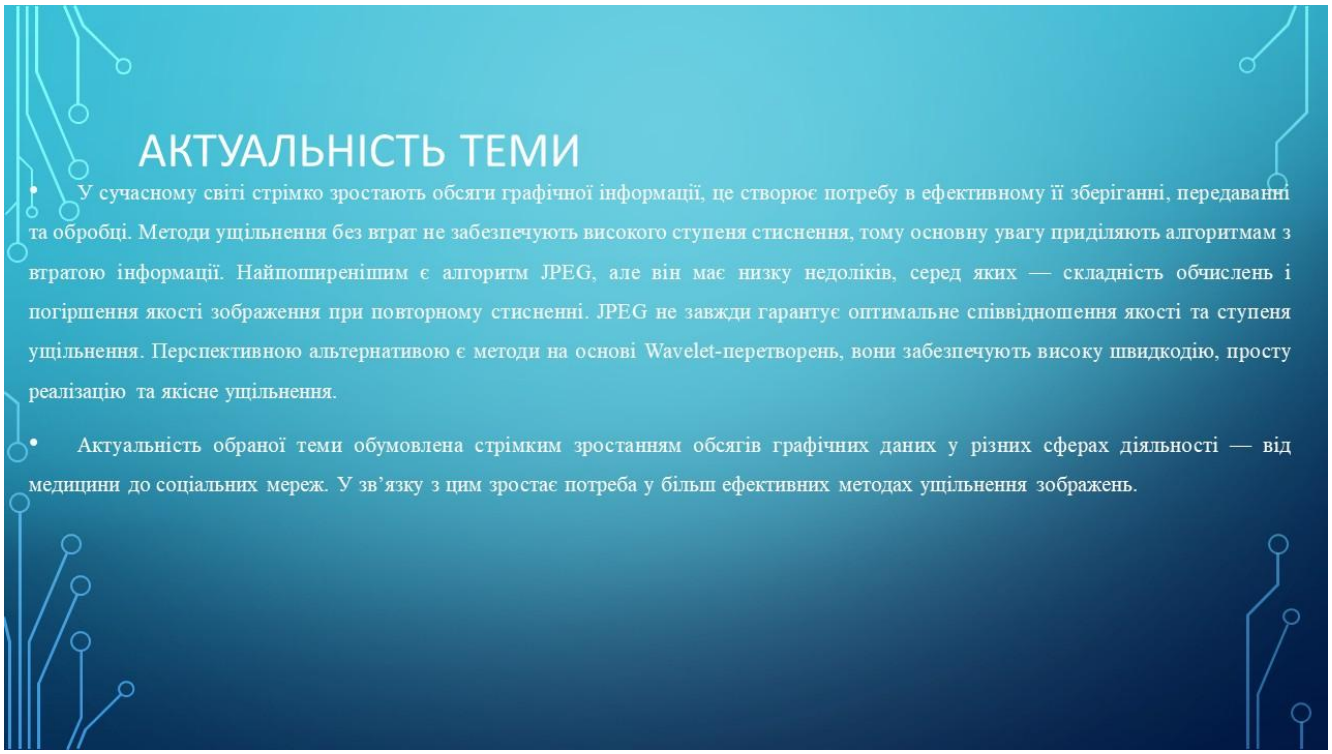


Рисунок Г.2 – Актуальність теми

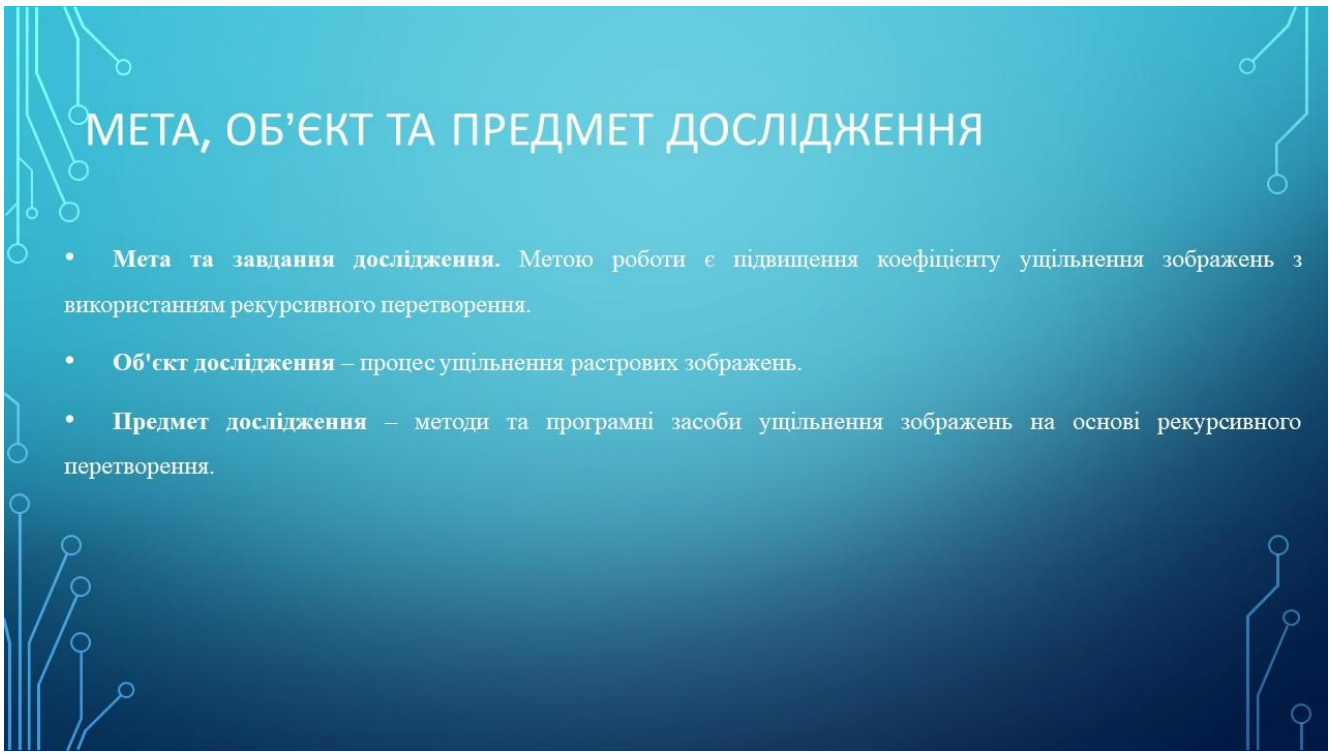


Рисунок Г.3 – Мета, об’єкт та предмет дослідження

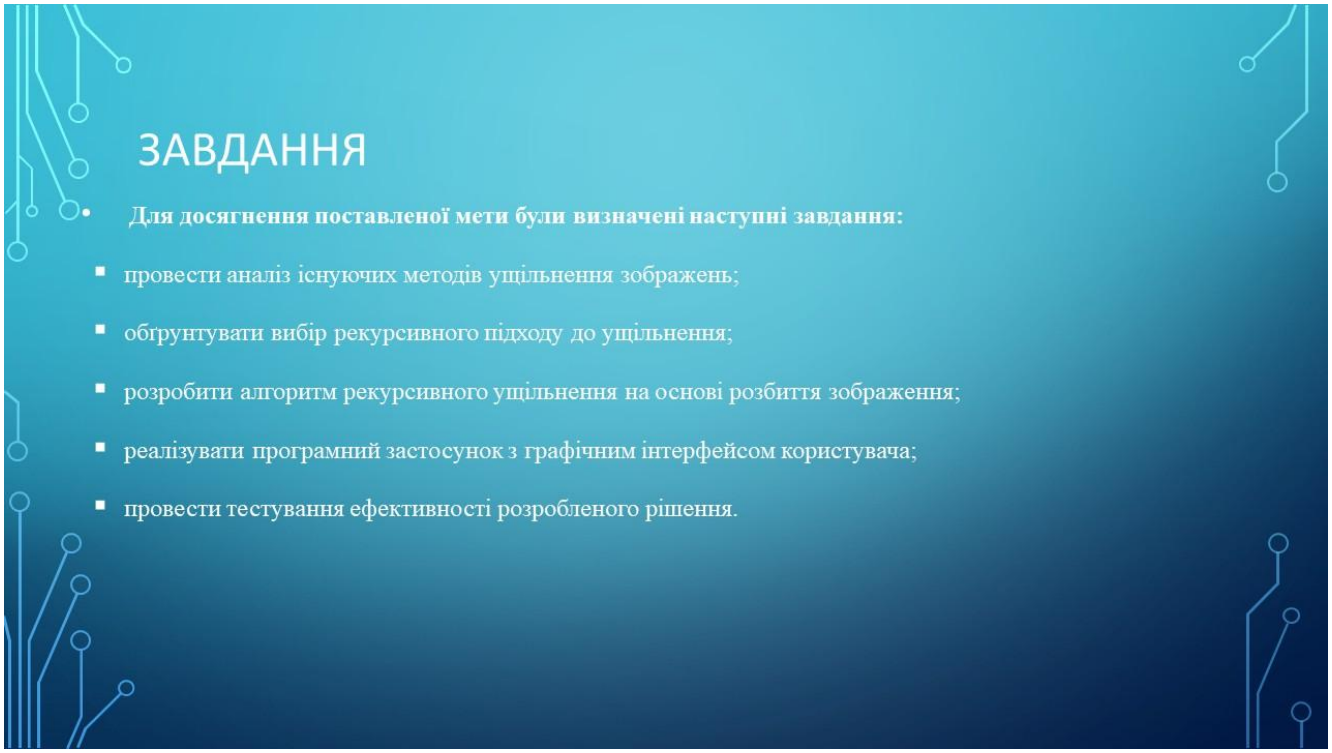


Рисунок Г.4 – Завдання

Рисунок Г.5 – Новизна

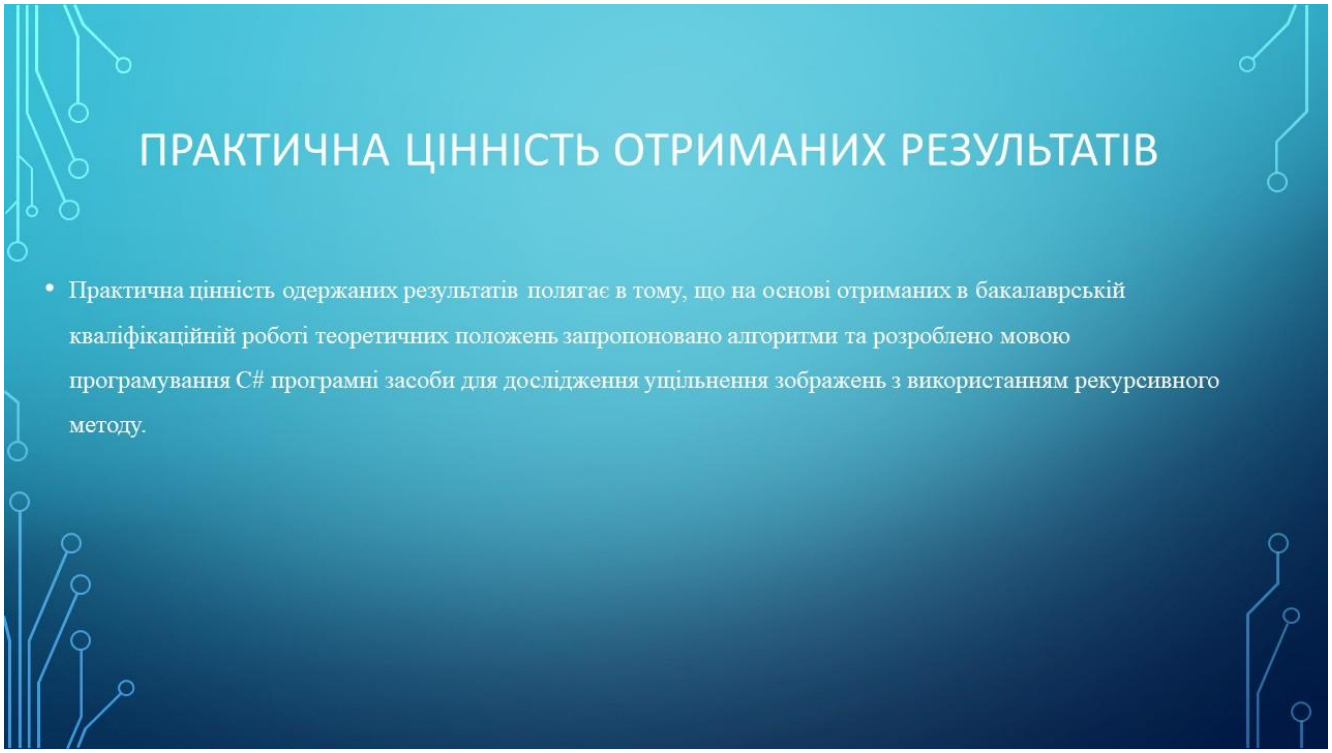


Рисунок Г.6 – Практична цінність отриманих результатів

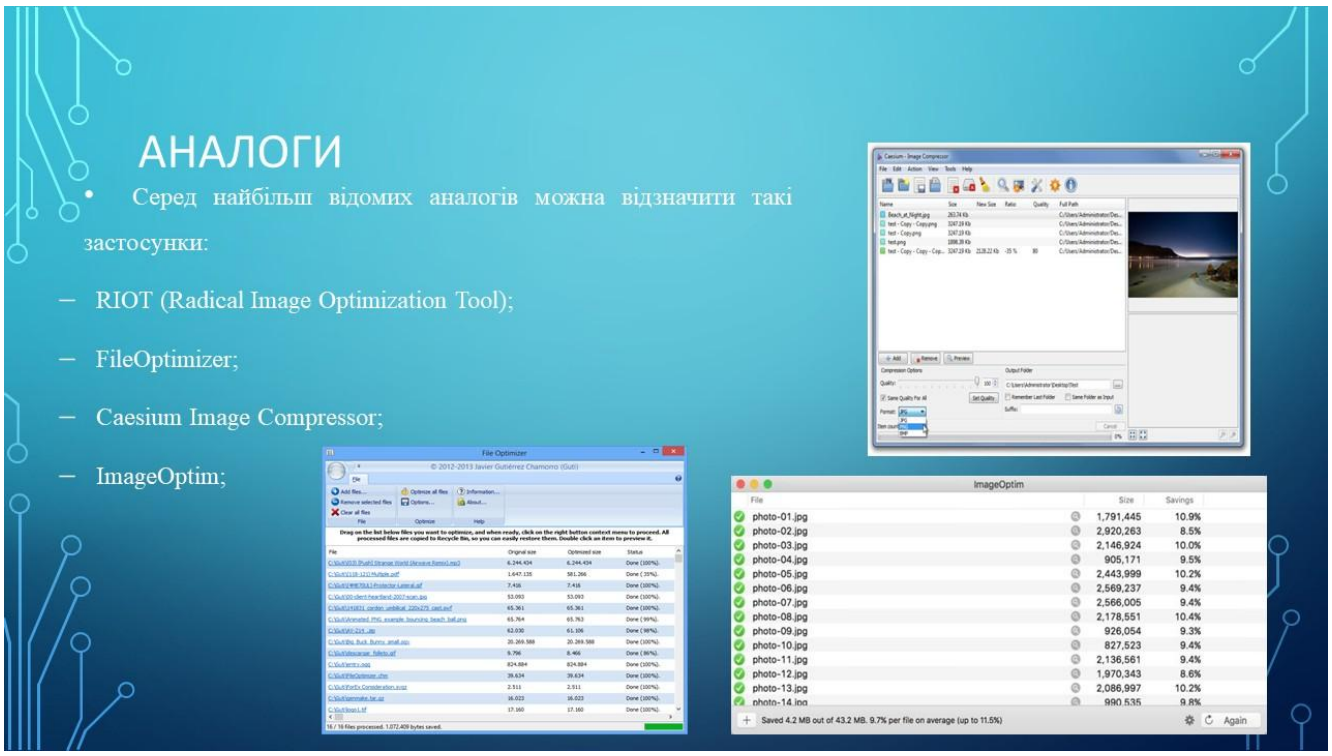


Рисунок Г.7 – Аналоги

Критерій	RIOT	File Optimizer	CIC	ImageOptim	Власна розробка
Простота інтерфейсу	+	+	+	-	+
Підтримка рекурсивної обробки	+	+	+	+	+
Адаптивне стиснення	-	-	-	-	+
Точний контроль якості	+	-	+	-	+
Швидкодія при обробці великих обсягів	+	-	+	+	+

Рисунок Г.8 – Порівняльний аналіз аналогів

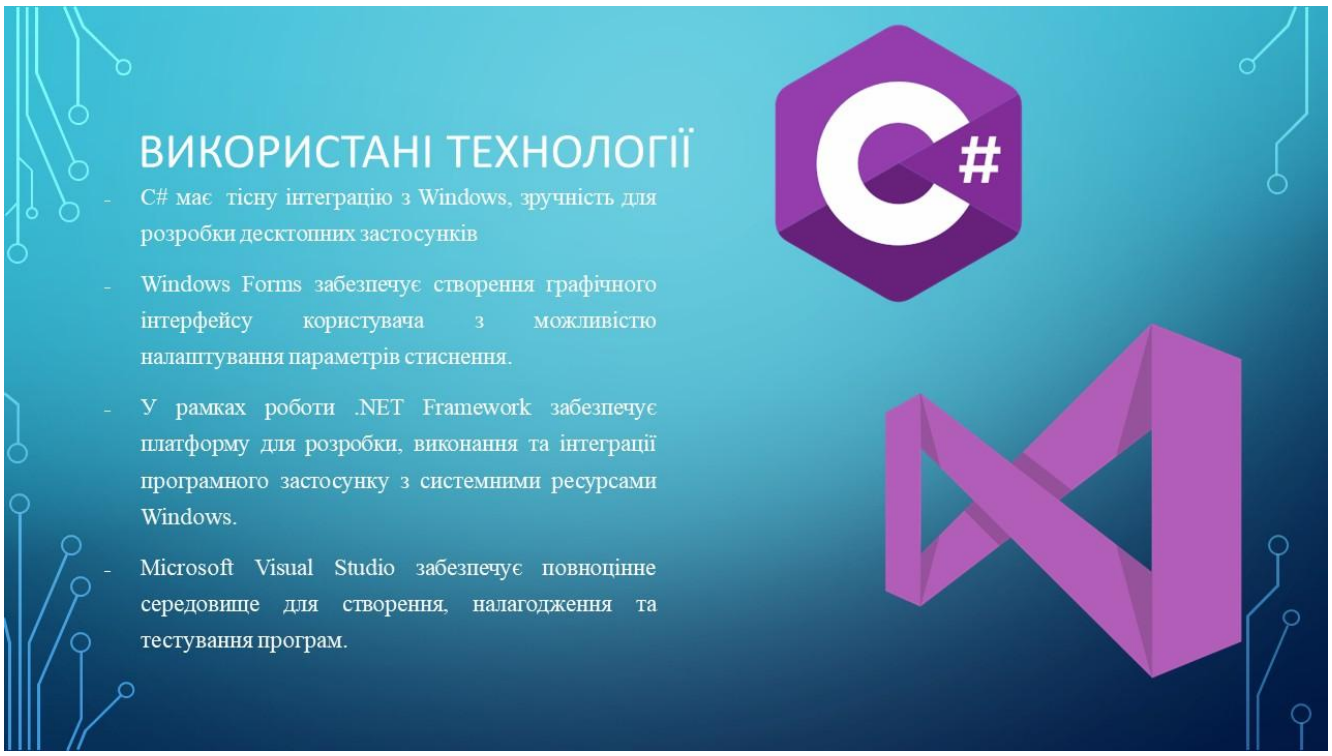


Рисунок Г.9 – Використанні технології

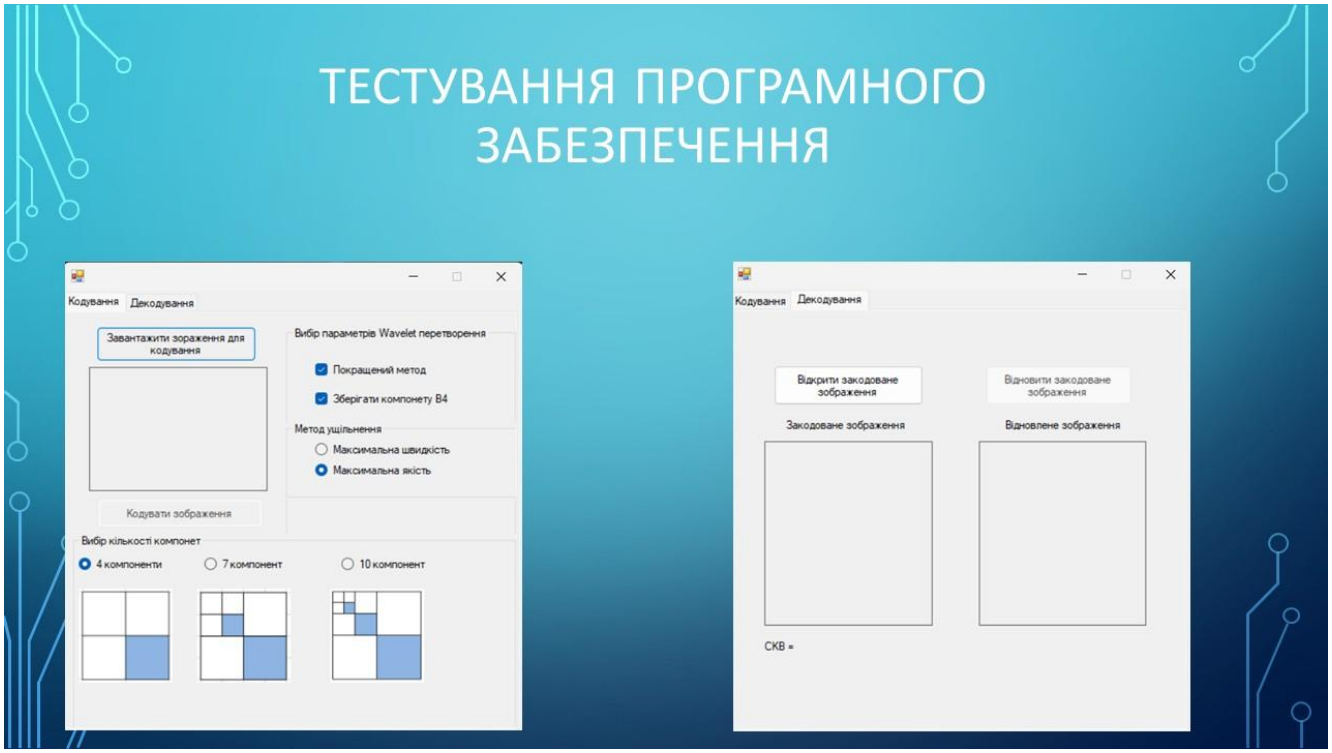


Рисунок Г.10 – Тестування програмного забезпечення

Тестування програми на прикладі файлу Lena

Файл та тип ущільнення	Тип зображення	Початковий розмір даних зображення, байт	Розмір після ущільнення, байт	Кількість компонент	Погіршення візуальної якості
Lena512.bmp	напівтонове	786572	177650	4	Не помітно
Lena512.bmp	напівтонове	786572	148519	10	Не помітно
Lena512.bmp (Jpeg)	напівтонове	786572	82912	-	Не помітно
lena512color.tiff	кольорове	786572	390990	4	Не помітно
lena512color.tiff (WH8)	кольорове	786572	296452	10	Не помітно
lena512color.tiff (Jpeg)	кольорове	786572	95646	-	Не помітно

Рисунок Г.11 – Тестування програми на прикладі файлу Lena

Приклади зображень

Рисунок Г.12 – Приклади зображень

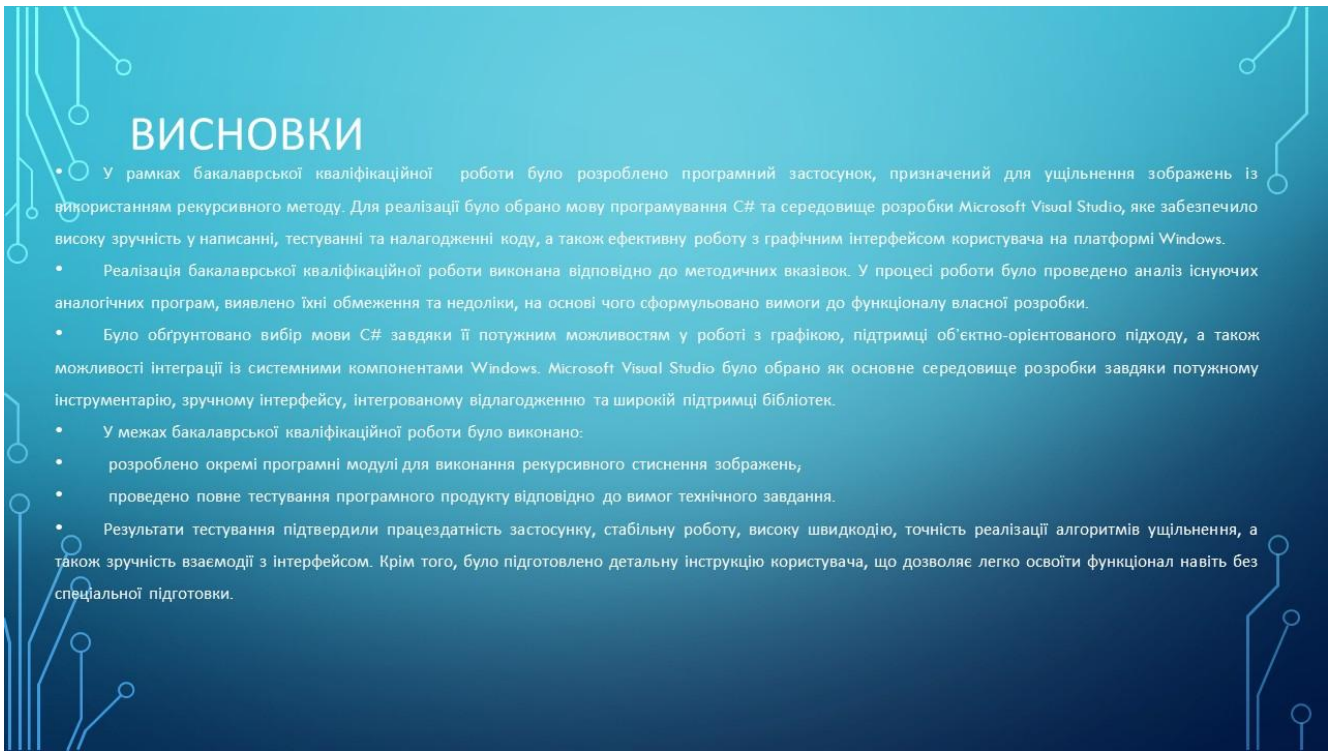


Рисунок Г.13 – Висновки

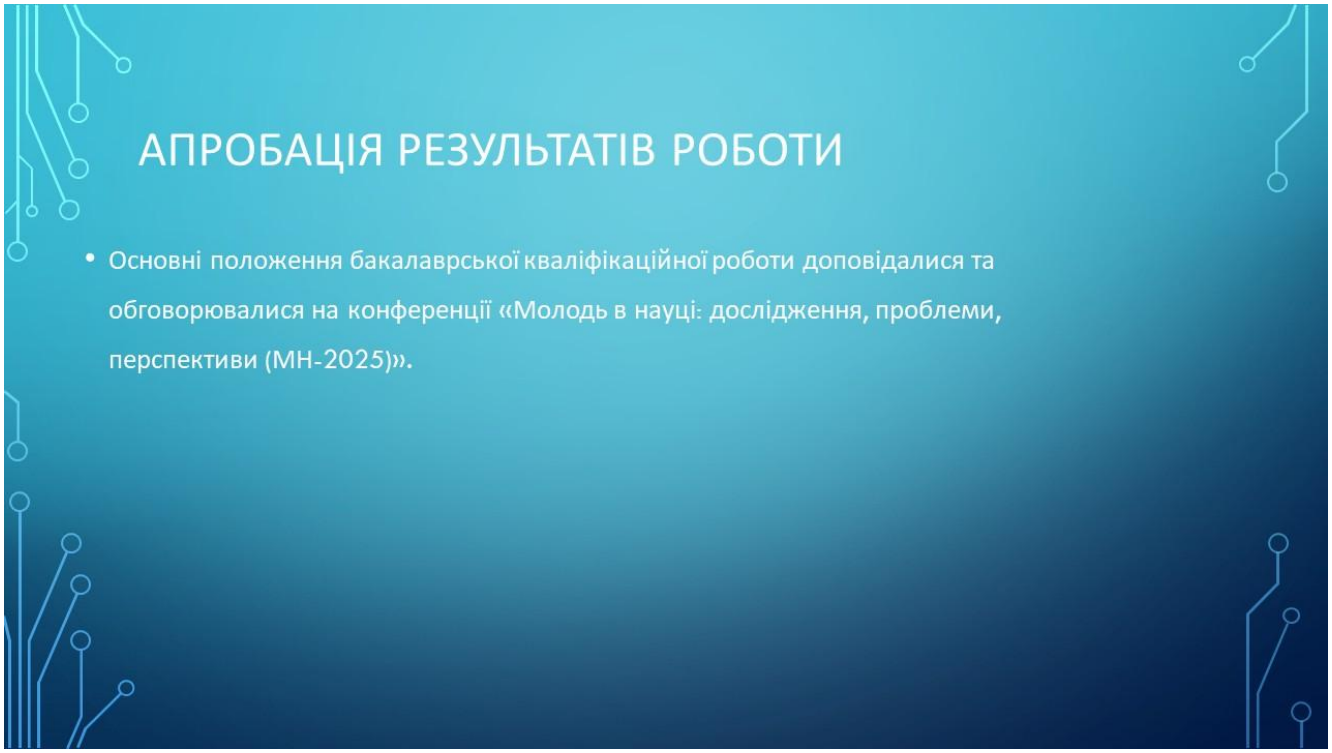


Рисунок Г.14 – Апробація

Рисунок Г.15 - Фінальний слайд