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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-1
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Молчанюк Є.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

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

Молчанюку Євгену Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень*

2. Керівник роботи *Коваленко анна Степанівна, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 32-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Молчанюк Є.В. Дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи стиснення зображень за допомогою вейвлет-перетворень.

Метою розробки є дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень.

Об'єктом дослідження є процес стиснення зображень за допомогою вейвлет-перетворень.

Предметом дослідження є методи стиснення зображень за допомогою вейвлет-перетворень.

Методи дослідження базуються на методах комп'ютерної графіки, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень.

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

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

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

Програму розроблено в середовищі Visual C#.

Ключові слова: комп'ютерні науки, стиснення зображень, вейвлет-перетворення

ABSTRACT

Molchaniuk E.V. Research and software implementation of the image compression system using wavelet transforms. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software is developed, which is intended for the image compression system using wavelet transforms.

The purpose of the development is the research and software implementation of the image compression system using wavelet transforms.

The object of research is the process of image compression using wavelet transforms.

The subject of research is image compression methods using wavelet transforms.

Research methods are based on computer graphics methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the image compression system using wavelet transforms.

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

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

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Visual C# environment.

Keywords: computer science, image compression, wavelet transform

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АЧХ	–	амплітудно-частотна характеристика
ВП	–	вейвлет-перетворення
ВЧ	–	високі частоти
ВФ	–	вейвлет-функції
ГА	–	генетичний алгоритм
ДПФ	–	дискретне перетворення Фур'є
НЧ	–	низькі частоти
УБК	–	метод усіченого блокового кодування
ФЧХ	–	фазочастотна характеристика

ВСТУП

Актуальність теми. У цей час методи цифрової обробки сигналів, що використовують у своїй роботі методи вейвлет-обробки, одержали широке поширення. Це пояснюється тими можливостями, які забезпечують вейвлет-функції, і в першу чергу, частотною й часовою локалізацією. Вейвлети й засновані на них вейвлет-перетворення були запропоновані на початку 1990-х років на основі модифікованих алгоритмів роботи з банками фільтрів [1-6] і в наступний час інтенсивно розвивалися. Великий внесок у розробку теоретичних основ вейвлетів внесли Мейер (Meyer), Добеши (Daubechies) і Маллат (Mallat), що опублікували перші теоретичні роботи в цьому напрямку. Основна маса книг і статей в області вейвлет-перетворень опублікована за рубежем. У Україні інтерес у вейвлетам активізувався трохи пізніше – у середині 1990-х років. Незважаючи на те, що основна робота Добеши (Daubechies) відноситься до початку 1990-х років, українською мовою вона вперше з'явилася тільки в 2001 році. Саме на цей часовий інтервал і доводиться публікація українською мовою ряду основних матеріалів по вейвлетам. Це перевідні роботи Чуй К., Уелстіда С., Блатера К. і роботи вітчизняних авторів (наприклад, Короновський А.). Але до виходу у світ зазначених фундаментальних українськомовних матеріалів, робота вітчизняних дослідників будувалася на базі закордонних матеріалів і невеликих вітчизняних статей. Особливо хочеться відзначити прекрасну оглядову статтю, орієнтовану на тих, хто тільки почав займатися цим предметом і які інтересуються його застосуванням, з демонстрацією вейвлет-перетворень деяких сигналів, опубліковану в журналі УФН в 1996 р. (автор Астафьева Н.М.), і яка викликала широкий інтерес до теми вейвлетів. У ній розглянуті безперервні вейвлет-перетворення, що дають наочне й видовищне подання результатів аналізу сигналу у вигляді локальних мінімумів і максимумів і скелетоних графіків вейвлет-коефіцієнтів. У цей час опубліковані сотні книг і тисячі статей по вейвлетам, що

відрізняються різним підходом до центральної теми. Інтерес викликає робота Перебіна А.В., де розкрита питання про систематизацію термінології вейвлет-перетворень. Вейвлет-фільтри являють собою окремий випадок реалізації банків фільтрів, тому розвиток теорії банків-фільтрів у цей час також визначає й розвиток напрямку вейвлетів. У зв'язку із цим хочеться відзначити внесок вітчизняних учених у розвиток цього напрямку (Миронов У.Г, Чобану М.К. і ін.)

При розгляді більшості алгоритмів, пов'язаних з вейвлетами, обробка одномірних сигналів і зображень зводиться до фільтрації. Теоретичні дослідження, що стосуються подання вейвлет-аналізу сигналів за допомогою процедури фільтрації з використанням звичайних цифрових фільтрів, проведені ще на початку 1990-х років. Наприклад, основним критерієм, якому повинен задовольняти цифровий фільтр, що претендує на використання при вейвлет-обробці – це властивість квадратурно-дзеркальності. Властивості й деякі питання синтезу квадратурно-дзеркальних фільтрів викладені в роботах. У зв'язку із простотою подання вейвлет-аналізу в рамках теорії цифрових фільтрів, і з огляду на великий обсяг матеріалу, накопичений з питань подання й синтезу одномірних і багатомірних цифрових фільтрів, у дисертації виклад матеріалу й проведення різних досліджень у рамках вейвлет-перетворень виконано з позицій цифрової фільтрації. При цьому використовувався матеріал по цифровій фільтрації, представлений у роботах Хеммінга Р. Каппеліні В., Гольденберга Л.М. і ін.

Одними з найважливіших об'єктів дослідження в даній роботі є параметризація й синтез одномірних і двовимірних вейвлет-функцій.

Наявність деяких ступенів волі при визначенні одномірного вейвлету (а особливо двовимірних вейвлетів) дозволяє розраховувати вейвлет-функції з різними наперед заданими частотними й часовими властивостями. Використання вейвлет-функції з різною частотною вибірковістю дозволяє розширити застосовність вейвлетів і використовувати їх у завданнях погодженої фільтрації, розпізнавання образів (наприклад, при розпізнаванні різних примітивів – дуг, ліній). У цей час інтерес до вейвлет-функцій зі специфічними властивостями зріс.

Це пояснюється розширенням області застосування вейвлетів і повнотою дослідження властивостей і застосувань класичних вейвлет-функцій.

Питання параметризації вейвлетів у літературі викладені з позицій параметризації банків фільтрів шляхом конкретизації. До основних робіт відносяться роботи Зоу (Zou H.) і Поллена (Pollen D.). Однак слід зазначити, що представлені роботи стосуються питань параметризації одномірних вейвлет-функцій. Так само до них можна звести параметризацію й роздільних двовимірних вейвлет-функцій, а для подання двовимірних нероздільних вейвлет-функцій потрібний уже інший математичний апарат. Питання параметризації двовимірних нероздільних вейвлет-функцій також є одним з об'єктів дослідження в дисертації. Основне використання їх – у завданнях обробки зображень.

Методи оптимізації вейвлет-функцій і адаптивного вибору коефіцієнтів вейвлет-розкладання є в завданнях стиску, які доповнюють друг друга. При використанні в розкладанні різних вейвлет-функцій, виходять різні набори коефіцієнтів вейвлет-перетворення й до кожного з наборів застосовні методи "оптимізації нуль-дерев". Основне питання адаптивного вейвлет-перетворення, розглянутий у дисертації – оптимізація частотної вибірковості ВФ. При стиску зображень за допомогою вейвлетів оптимізація частотної вибірковості вейвлет-фільтру є первинною стадією обробки, наступні стадії – відбір і нерівномірне квантування коефіцієнтів розкладання – спрямовані на оптимізацію подання результатів першої стадії. Тому в першу чергу робота спрямована на оптимізацію вейвлет-функцій і набір результатів, що представляються, орієнтований на зіставлення з результатами обробки, отриманими при використанні інших методів оптимізації.

Один з напрямків вейвлет-обробки пов'язане з розпізнаванням образів. У рамках даного напрямку також представлені роботи, пов'язані з оптимальним вибором вейвлет-фільтрів під різні критерії. Наприклад, у роботі [2] проведене дослідження впливу різних критеріїв вейвлет-фільтру – гладкості, лінійності фазочастотної характеристики (ФЧХ) – на точність розпізнавання текстури

зображення й у рамках конкретного критерію представлений вибір оптимального вейвлет-фільтру. У рамках цього напрямку хочеться відзначити роботи Донохо [3, 5-6], що є продовженням теорії вейвлет-перетворення й присвячені синтезу особливих функцій бімлетів (Beamlets), ріджлетів (Ridglets), що відрізняються лінійною частотно-часовою локалізацією й ортогональністю розкладання. Основна перевага їх досягається в завданнях розпізнавання ліній, дуг у зображеннях з високим рівнем шуму (особливо якщо потужність шуму в рази перевищує потужність зображення).

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень.

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

– Огляд існуючих систем стиснення зображень за допомогою вейвлет-перетворень.

– Дослідження системи стиснення зображень за допомогою вейвлет-перетворень.

– Програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень.

Об'єктом дослідження є процес стиснення зображень за допомогою вейвлет-перетворень.

Предметом дослідження є методи стиснення зображень за допомогою вейвлет-перетворень.

Методи дослідження базуються на методах комп'ютерної графіки, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод стиснення зображень за допомогою вейвлет-перетворень.

– Розроблено вітчизняний продукт стиснення зображень за допомогою вейвлет-перетворень, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі стиснення зображень за допомогою вейвлет-перетворень.

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

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для стиснення зображень за допомогою вейвлет-перетворень. Бажаєте стиснути зображення перед завантаженням в Інтернет? Оптимізація зображень для веб-сайту фотографій допоможе швидше завантажувати ваш сайт. І швидша швидкість завантаження означає кращий рейтинг у пошукових системах.

Багато програм для редагування фотографій, наприклад Adobe Photoshop, оптимізують зображення. Але, залежно від вашого бюджету, ці програми можуть бути витратами, які ви не можете собі дозволити. На щастя, є різноманітні онлайн-інструменти, якими можна користуватися безкоштовно та які забезпечують дивовижні результати.

Чому потрібно оптимізувати зображення? Повнорозмірні зображення збільшують час завантаження вашої веб-сторінки, що дратує ваших користувачів. Насправді дослідження показали, що користувачі переходять зі сторінок, якщо вони завантажуються більше 3 секунд. Оптимізація ваших зображень значно допоможе пришвидшити роботу веб-сайту, щоб ви могли утримувати користувачів на своєму сайті.

1.2 Область застосування

Областю застосування системи є стиснення зображень. Метою стиснення зображень є зменшення нерелевантності та надмірності даних зображення, щоб мати можливість зберігати або передавати дані в ефективній формі. Це стосується мінімізації кількості бітів, необхідних для представлення зображення. Стиснення зображення може бути з втратами або без втрат. Для архівування

краще використовувати стиснення без втрат часто для медичних зображень, технічних креслень, графічних зображень або коміксів. Методи стиснення з втратами, особливо коли вони використовуються на низьких бітрейтах, створюють артефакти стиснення. Методи з втратами особливо підходять для природних зображень, таких як фотографії, у програмах, у яких незначна (іноді непомітна) втрата точності прийнятна для досягнення суттєвого зниження швидкості потоку. Стиснення з втратами, яке створює непомітні відмінності, можна назвати візуальним без втрат.

Крім цих трьох методів, фактично, ми можемо розглядати процес вбудовування інформації в зображення обкладинки як особливий вид цифрової обробки зображень, оскільки його вхід і вихід є цифровими зображеннями.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

1. JPEG Optimizer

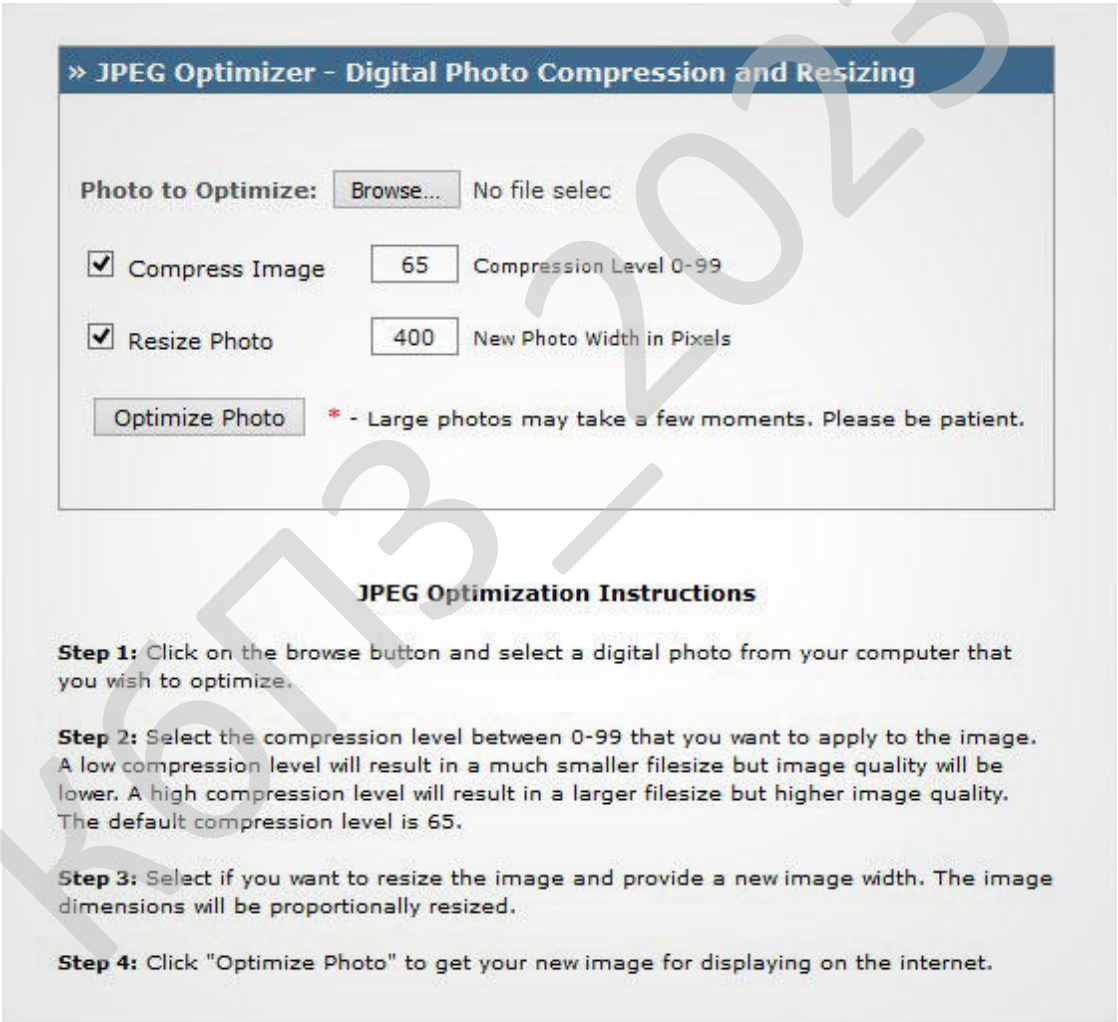

Рисунок 2.1 – JPEG Optimizer

JPEG Optimizer дозволяє завантажувати та стискати фотографії в Інтернеті. Цей простий інструмент працює, як випливає з його назви, лише з файлами JPEG.

Що чудово в JPEG Optimizer, так це те, що він дозволяє змінювати розміри зображень перед оптимізацією. Зміна розміру зображень заощадить час завантаження, а з цією онлайн-платформою вам не доведеться розділяти робочий процес на два етапи.

JPEG Optimizer також дозволяє вибрати власний рівень оптимізації, що дозволяє контролювати якість оптимізованого зображення. Ця функція особливо важлива для фотографів, оскільки ви захочете знайти оптимальне співвідношення між збереженням якості та економією місця.

2. Optimizilla

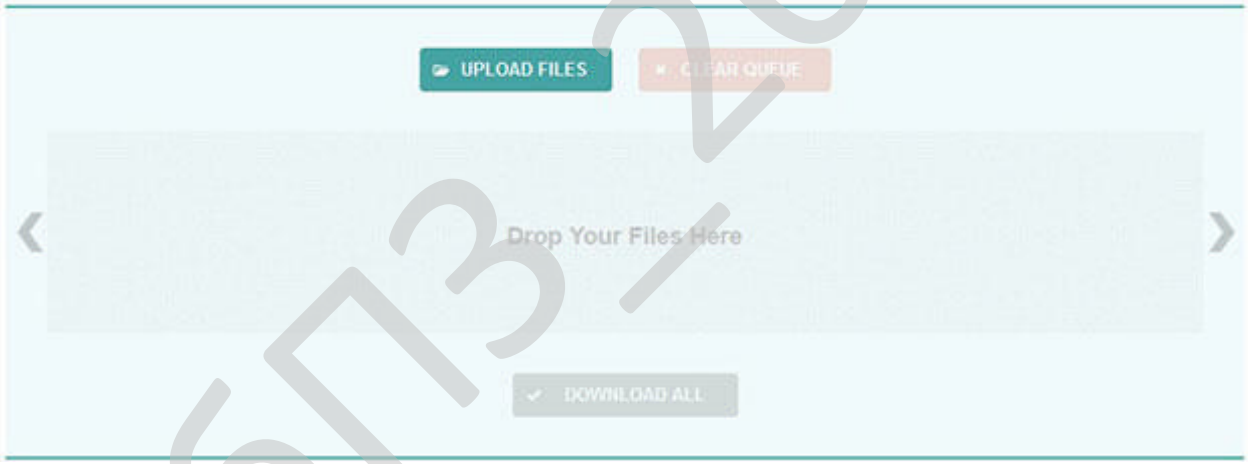

Рисунок 2.2 – Optimizilla

Optimizilla вдається відтворити чудову якість ваших зображень із мінімально можливим розміром файлу. За допомогою цього інструменту ви можете стискати фотографії у форматі JPEG і PNG.

Optimizilla також має повзунок, який показує версію до та після вашої фотографії. Таким чином ви зможете попередньо переглянути якість свого зображення, перш ніж продовжити. За допомогою повзунка ви можете вирішити,

наскільки ви хочете оптимізувати своє зображення, перш ніж відчуєте помітну втрату якості.

Цей онлайн-інструмент чудово підходить для пакетної обробки. Optimizilla дозволяє завантажувати до 20 зображень одночасно та встановлювати рівень стиснення для кожної фотографії окремо.

На жаль, оскільки Optimizilla працює лише з файлами JPEG і PNG, ви не зможете використовувати її для оптимізації PDF-файлів. Деякі інші в цьому списку краще підходять для оптимізації PDF.

3. Kraken.io

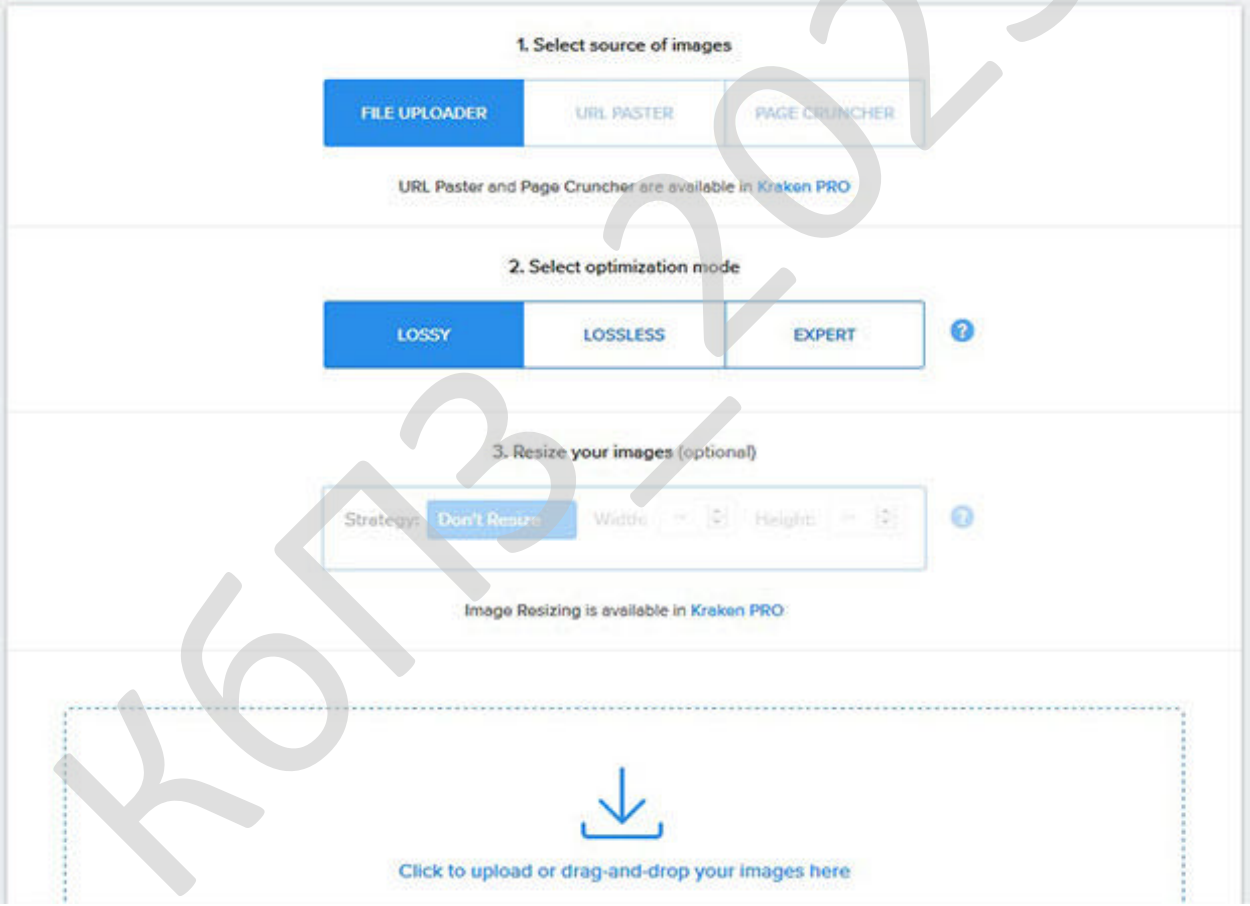

Рисунок 2.3 – Kraken.io

Kraken.io – ще один оптимізатор масових зображень. За допомогою Kraken.io ви можете оптимізувати велику кількість файлів JPEG, PNG і

анімованих GIF.

На відміну від інших оптимізаторів зображень у цьому списку, Kraken.io оптимізує ваші файли для найменшого розміру. Тобто за допомогою Kraken.io ви завжди отримуватимете версію введеного вами зображення з найменшим розміром файлу. Потім ви можете завантажувати стислі фотографії по одній або у форматі.zip.

Kraken.io також дозволяє експортувати файли в Dropbox або імпортувати файли з Box, Dropbox або Google Drive.

Якщо ви шукаєте розширені функції, Kraken.io пропонує професійну версію, яка має низку інших переваг, як-от можливість зміни розміру зображення, можливість одночасно вводити кілька джерел зображень, завантажувати зображення необмеженого розміру тощо.

Безкоштовна версія дозволяє стискати файли фотографій розміром до 32 МБ кожен і загальним розміром фотографій до 100 МБ. Отже, наскільки добре Kraken.io працюватиме для вас, залежить від початкового розміру зображень, які ви хочете оптимізувати.

4. Optimole

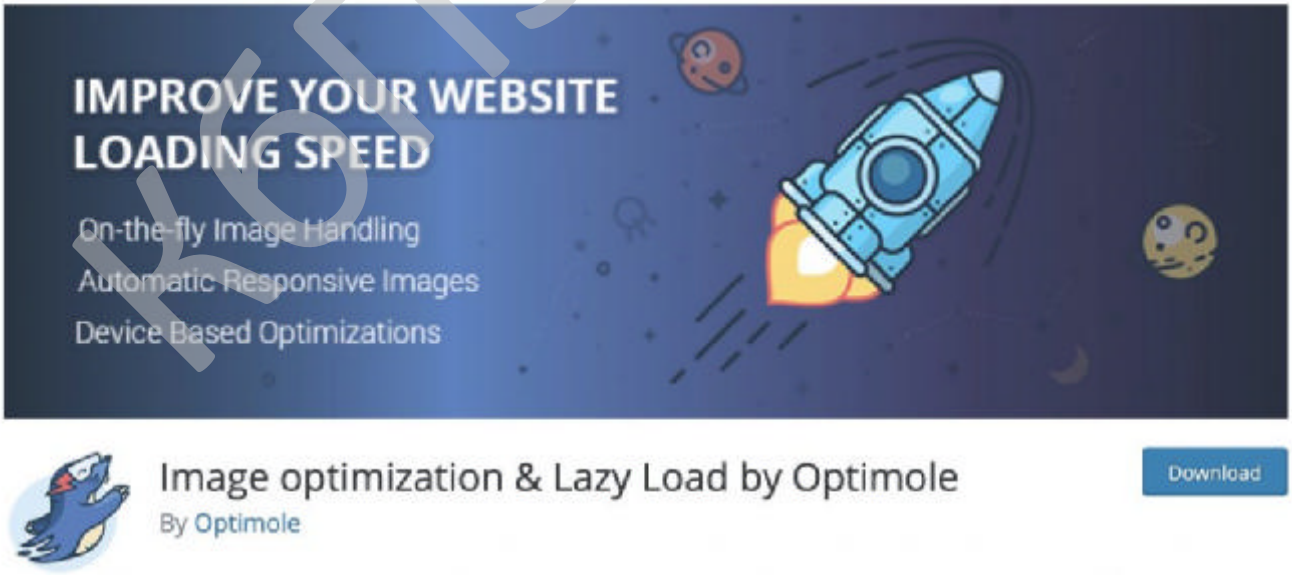

Рисунок 2.4 – Optimole

Optimole – це популярний плагін для стиснення та оптимізації зображень, створений тими ж людьми, що стоять за ThemeIsle.

Він може стискати, змінювати розмір, відкладено завантажувати та розміщувати ваші зображення на CDN.

Звичайно, послуги SAAS не безкоштовні, але є безкоштовний план Optimole, щоб ви могли розпочати роботу з їх системою.

Найкраща частина полягає в тому, що ви можете встановити плагін WordPress на свій сайт і легко оптимізувати зображення без особливих зусиль.

Нехай це крутиться за лаштунками.

5. ImageRecycle

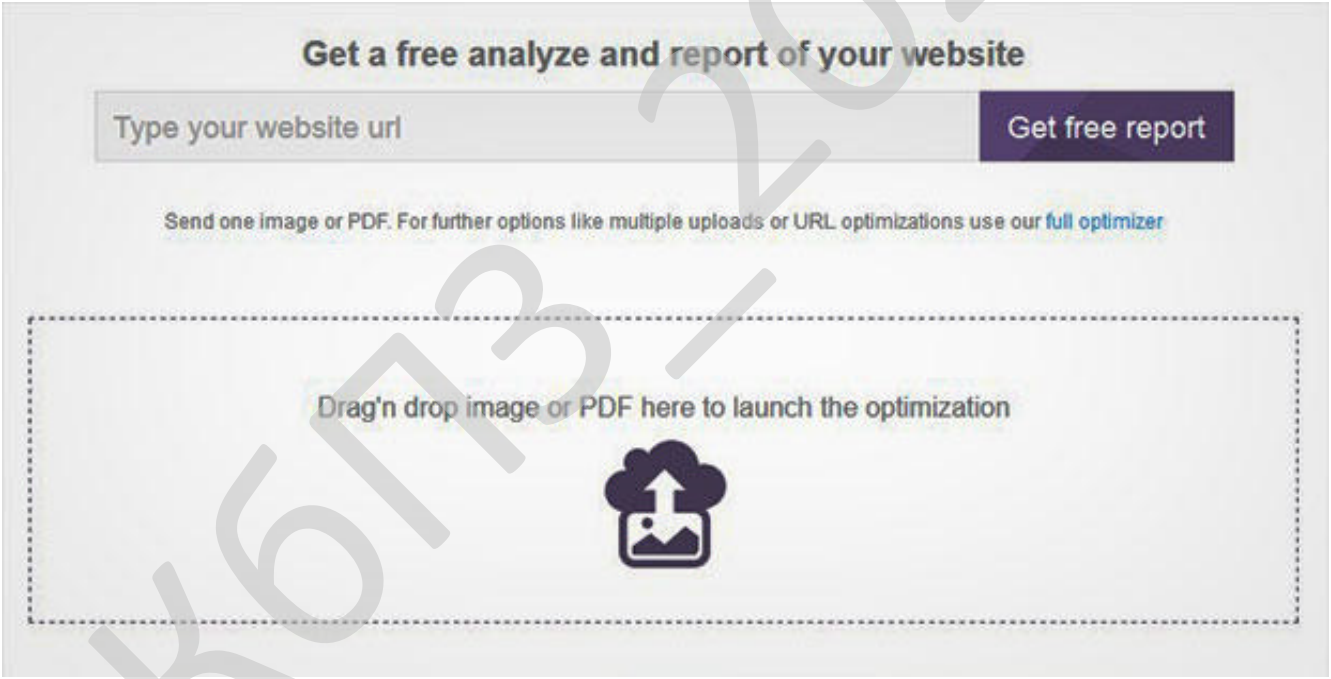

Рисунок 2.5 – ImageRecycle

ImageRecycle стискає ваші зображення JPEG, PNG, GIF і PDF із вражаючими результатами. Їх інтерфейс перетягування дозволяє перетягувати фотографії з робочого столу в оптимізатор для полегшення робочого процесу.

Сайт також пропонує безкоштовний аналіз веб-сайту. Коли ви надсилаєте

запит на аналіз, ImageRecycle повертає безкоштовний звіт, розроблений, щоб допомогти вам зрозуміти, які зображення на вашому сайті виграють від оптимізації для підвищення ефективності вашого сайту.

ImageRecycle має плагін WordPress, додаток Shopify, розширення Joomla та розширення Magento. Якщо ви користуєтеся будь-яким із цих інших сайтів, ви можете використовувати розширення ImageRecycle для стискання зображень замість переходу на окремий сайт.

З ImageRecycle ви отримуєте доступ до 15-денної безкоштовної пробної версії з обмеженням у 100 Мб. Після цього ви можете придбати платний план – 10 доларів США за 1 ГБ, 20 доларів США за 3 ГБ і 50 доларів США за 10 ГБ.

6. CompressNow

Рисунок 2.6 – CompressNow

CompressNow – ще один простий у використанні інструмент оптимізації, який дозволяє масове завантаження. Просто завантажте зображення JPEG, PNG і

GIF зі свого комп'ютера, установіть відсоток стиснення для зображень і завантажте їх для використання на своєму сайті.

Ви можете перетягувати до 10 зображень одночасно. Але, на відміну від Optimizilla, CompressNow застосовує один рівень оптимізації до всіх фотографій, які ви завантажуєте. Якщо всі вони потребують однакового рівня стиснення, то CompressNow спрощує вашу роботу!

7. Trimage

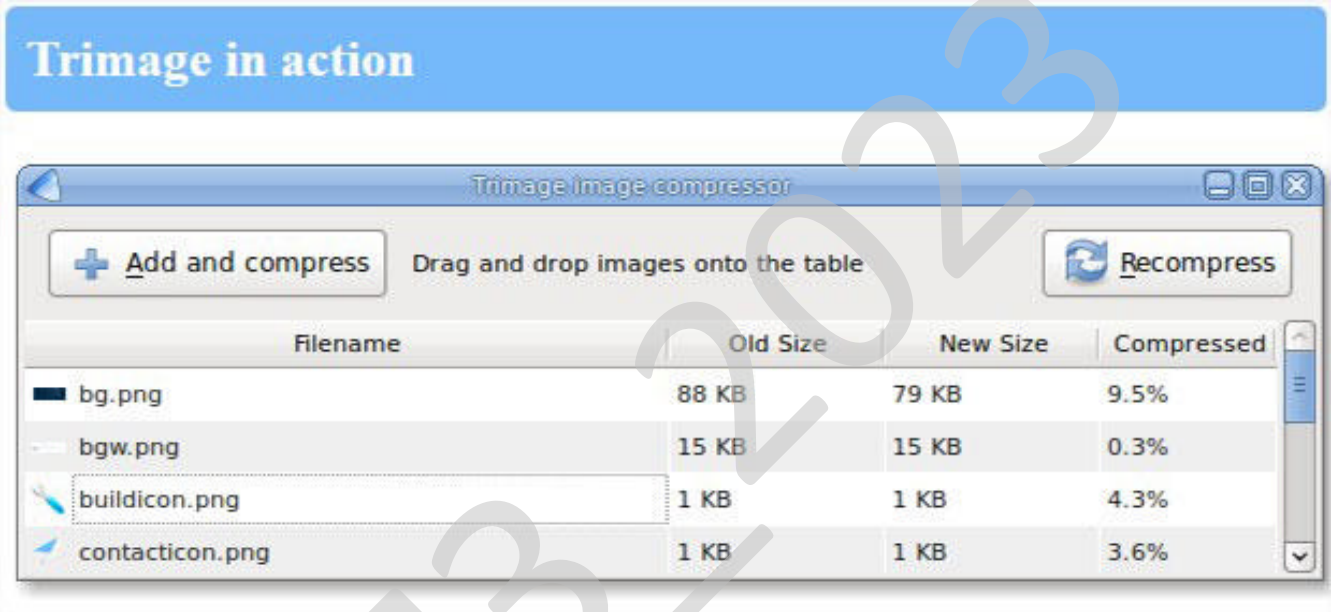

Рисунок 2.7 – Trimage

Trimage добре працює для користувачів Linux для видалення EXIF і метаданих із зображень. Потім цей інструмент стиснення зображень стискає ваші зображення JPEG і PNG до найвищого можливого рівня.

Якщо ви користуєтеся Mac або ПК, інші варіанти в цьому списку будуть кращим вибором для вашої системи. Trimage найкраще підходить для користувачів Linux.

8. Online Image Optimizer

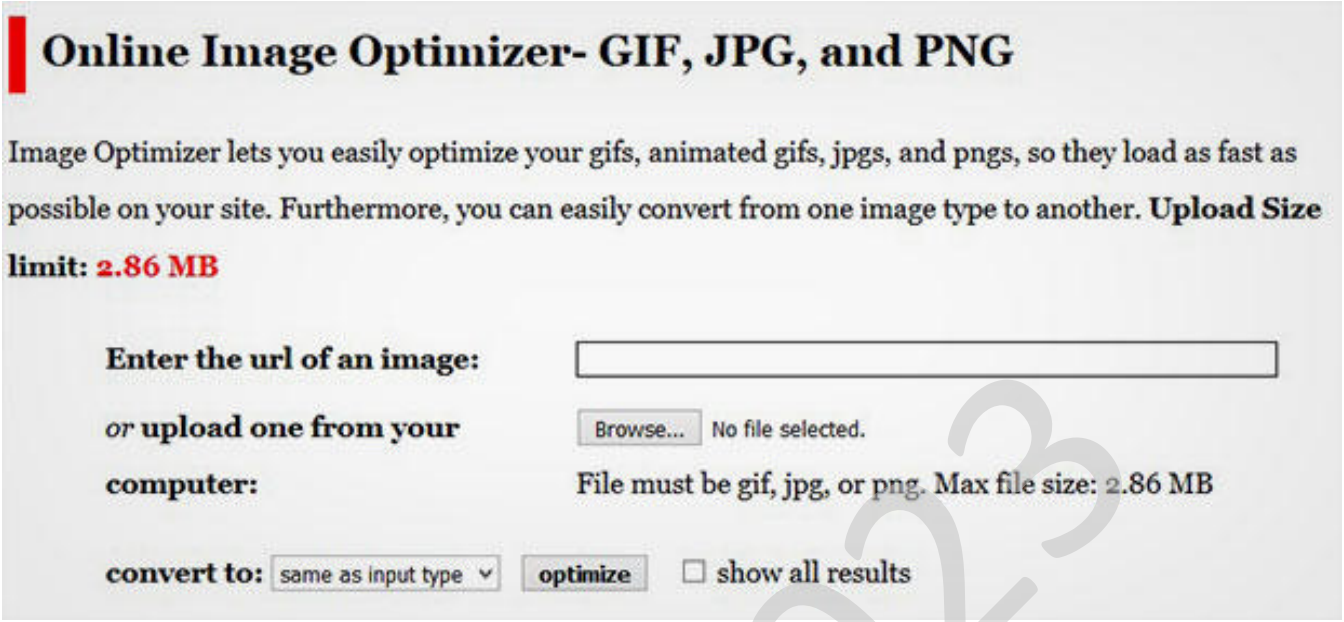

Рисунок 2.8 – Online Image Optimizer

Онлайн-оптимізатор зображень, продукт Dynamic Drive, допомагає легко оптимізувати зображення всіх основних форматів, щоб ваші фотографії швидше завантажувалися на веб-сайт.

Ці інструменти також містять функції перетворення зображень, які можуть допомогти вам перетворити ваші JPEG на PNG для використання в Інтернеті.

На відміну від деяких інших оптимізаторів зображень, Online Image Optimizer також дозволяє завантажувати зображення за допомогою URL-адреси. Це означає, що ви можете оптимізувати зображення, які вже є на вашому веб-сайті! У довгостроковій перспективі це економить ваш час і зусилля на заміну старого зображення оптимізованою версією онлайн.

Онлайн-оптимізатор зображень дозволяє завантажувати лише одне зображення за раз, і це зображення може мати максимальний розмір файлу 2,86 МБ.

9. Tiny PNG

Рисунок 2.9 – Tiny PNG

Tiny PNG є одним із найстаріших і найпопулярніших безкоштовних інструментів оптимізації зображень. Завдяки безлічі можливостей для стиснення зображень для вашого сайту, цей онлайн-інструмент чудово працює з файлами зображень JPEG і PNG.

Це дозволяє завантажувати до 20 зображень за один раз і до 100 зображень на місяць. Розмір зображення для кожного зображення не може перевищувати 5 Мб, але для більшості це не буде проблемою. Після стиснення ви можете завантажити стислі зображення на свій комп'ютер або легко зберегти їх у Dropbox.

Tiny PNG також пропонує плагін WordPress і розширення Magento, тож вам не потрібно переходити з власного веб-сайту, щоб стиснути зображення.

10. Resize Photos

Resize Photos – ще один безкоштовний інструмент стиснення зображень для оптимізації зображення. Ви також можете використовувати Resize Photos, щоб змінити розмір своїх зображень для використання на своєму сайті, в електронних листах або на форумах.

Просто завантажте зображення на онлайн-сторінку та встановіть рівень стиснення від 1 до 100. Потім завантажте стислі фотографії!

Ви також можете використовувати їхній сайт, щоб застосувати ефекти зображення з їхньої галереї під час процесу оптимізації. Ці ефекти включають підписи, фотоефекти, рамки, відображення, тіні та закруглені кути.

11. GiftOfSpeed

GiftOfSpeed пропонує безліч інструментів для стиснення для оптимізації ваших зображень PNG і JPEG. Він застосовує кілька методів стиснення зображень, щоб мінімізувати розміри файлів до найменшого можливого розміру.

За допомогою GiftOfSpeed ви також можете змінювати розміри зображень для кращої продуктивності онлайн.

Недоліком цього інструменту є те, що ви можете завантажити кілька файлів лише для оптимізації PNG, а не для оптимізації JPG.

12. Compressor.io

Compressor.io гарантує, що ви не втратите якість зображення, а також досягнете високого рівня стиснення.

Це дивовижний інструмент, відомий своєю гнучкістю для легкої оптимізації файлів JPEG, PNG, GIF і SVG. Compressor.io має можливість зменшувати розміри файлів до 90% і більше!

І знову падіння Compressor.io пов’язане з його обмеженнями на завантаження. Ви не можете використовувати Compressor.io для оптимізації кількох файлів зображень одночасно.

13. JPEGmini

JPEGmini відправляє ваші зображення на дієту! Він зменшує розмір файлу ваших зображень, тож ви можете насолоджуватися найкращою швидкістю завантаження зображень в Інтернеті. Він має кращий інтерфейс, ніж деякі інші варіанти, з можливістю оновлення до версії Pro.

Завантажте своє зображення, а потім скористайтеся повзунком, щоб переглянути версії до та після вашої фотографії. Ви побачите, що JPEGmini

зберігає якість зображення після стиснення.

14. Convert Image

Convert Image стискає зображення JPEG на ходу. Цей онлайн-інструмент оптимізації зображень дозволяє конвертувати фотографії JPEG в інші формати зображень, наприклад BMP, GIF, ICO, PNG тощо. Він навіть має деякі вбудовані налаштування, які можуть перевернути ваше зображення, обрізати, вирівняти зображення тощо.

15. PNGGauntlet

PNGGauntlet працює спеціально з файлами зображень PNG. Це програмне забезпечення, яке можна завантажити, поєднується з декількома компресорами, такими як PNGOUT, OptiPNG і DeflOpt, щоб стискати ваші зображення, не впливаючи на якість зображення. Він також працюватиме для перетворення та стиснення файлів JPG, GIF, TIFF і BMP у PNG.

З іншого боку, для оптимізації потрібно трохи більше часу, ніж для деяких інших варіантів.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

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

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

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

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

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

– Атрибути з декларативними метаданими про типи під час виконання.

– Вбудовані коментарі XML-документації.

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

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

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

Архітектура платформи .NET Framework

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

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

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

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи стиснення зображень за допомогою вейвлет-перетворень.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Вейвлети – це функції, які дозволяють аналізувати дані сигналів або зображень відповідно до масштабів або роздільної здатності. Обробка сигналів за допомогою хвильового алгоритму фактично працює так само, як і людина; або як цифрова камера обробляє візуальні шкали роздільної здатності та проміжні деталі. Але той самий принцип також фіксує сигнали мобільного телефону та навіть оцифровані кольорові зображення, які використовуються в медицині.

Хвильові сигнали реально використовуються в цих областях, наприклад, для апроксимації даних із різкими розривами, такими як уривчасті сигнали, або зображення з великою кількістю країв.

Хоча вейвлети є, можливо, одним із розділів теорії функцій, ми показуємо, що алгоритми, які дають результат, є ключем до обробки чисел, точніше, оцифрованої інформації, сигналів, часових рядів, нерухомих зображень, фільмів, кольорових зображень тощо.

Таким чином, застосування ідеї вейвлету включає велику частину обробки сигналів і зображень, стиснення даних, кодування відбитків пальців і багато інших галузей науки та техніки.

Ця робота зосереджена на обробці кольорових зображень з використанням спеціально розроблених вейвлет-алгоритмів і математичних порогових фільтрів. Незважаючи на те, що нещодавно було опубліковано чимало статей про операторську теорію вейвлетів, існує потреба в роботі, який пояснює деякі прикладні тенденції з нуля для теоретиків-операторів. Вейвлети є дуже міждисциплінарним предметом і він черпає ключові шляхи ідей із зовнішнього світу. Ми прагнемо окреслити різноманітні зв’язки між геометрією простору Гільберта та обробкою зображень.

Як працюють вейвлети в обробці зображень, аналогічно тому, як працюють наші очі. Залежно від місця спостереження можна по-різному сприймати ліс. Якщо ліс спостерігати з вершини хмарочоса, він буде спостерігатися як зелений; якщо це було помічено в машині, що рухається, це буде спостерігатися як дерева в лісі, що блимають, таким чином дерева тепер розпізнаються. Тим не менш, якщо це спостерігатиме хтось із нових, хто насправді ходить навколо нього, тоді можна буде спостерігати більше деталей дерев, таких як листя та гілки, і, можливо, сім мавп на вершині кокосової пальми. Крім того, витягнувши збільшувальне скло, можна навіть спостерігати текстуру дерев та інші дрібні деталі, які не можуть сприйняти голі люди .

WaveletImageProcessing дає змогу комп'ютерам зберігати зображення в багатьох масштабах роздільної здатності, таким чином розкладаючи зображення на різні рівні та типи деталей і наближаючи його до різних значень роздільної здатності. Отже, роблячи можливим масштабування для отримання більшої кількості деталей дерев, залишають і навіть мавпу на вершині дерева. Wavelets дозволяють одному стискати зображення, використовуючи менше пам'яті. простір із більшою кількістю деталей зображення.

Перевага розкладання зображень на наближені та деталізовані частини полягає в тому, що можна виділяти та маніпулювати даними з певними властивостями. Завдяки цьому можна визначити, чи потрібно зберігати більш конкретні деталі. Наприклад, зберігати більше вертикальних деталей замість того, щоб зберігати всі горизонтальні, діагональні та вертикальні деталі зображення, яке має більше вертикальних аспектів. Це дозволило б зображенню втратити певну кількість горизонтальних і діагональних деталей, але не вплинуло б на людське сприйняття зображення.

Як математично показано зображення можна розкласти на наближені, горизонтальні, вертикальні та діагональні деталі. Виконано N рівнів розкладання. Після цього на складеному зображенні виконується квантування, де на різних компонентах може бути виконане квантування, таким чином максимізуючи

кількість необхідних деталей та ігноруючи «небажані» деталі. Це робиться за допомогою порогового значення, коли деякі значення коефіцієнтів для пікселів із зображень «викидаються» або встановлюються на нуль, або на матриці зображення використовується певний ефект «згладжування». Цей процес використовується в JPEG2000.

Весь процес стиснення хвильового зображення виконується наступним чином: комп'ютер отримує вхідне зображення, виконується пряме вейвлет-перетворення на цифровому зображенні, виконується порогове значення для цифрового зображення, ентропійне кодування виконується на зображенні, де це необхідно, тому стиснення зображення виконується на комп'ютері. Потім зі стисненим зображенням виконується реконструкція вейвлет-перетвореного зображення, а потім На зображенні виконується зворотне хвильове перетворення, таким чином зображення реконструюється. У деяких випадках використовується алгоритм нульового дерева [Sha93], і відомо, що він має краще стиснення за допомогою алгоритму нульового дерева, але тут його не було реалізовано.

3.2 Розробка структурної схеми

Стиснення зображення на основі вейвлет-перетворення полягає у застосуванні вейвлет-алгоритму до декомпозиції зображення з різною роздільною здатністю та досягненні стиснення зображення шляхом кодування отриманих вейвлет-коефіцієнтів. Спочатку на зображенні виконується багаторівнева вейвлет-розкладка для отримання відповідних вейвлет-коефіцієнтів. Потім кожен шар вейвлет-коефіцієнтів квантується, щоб отримати об'єкти квантованих коефіцієнтів. Нарешті, об'єкти квантованих коефіцієнтів кодуються для отримання результатів стиснення. У цьому документі використовується вейвлет Хаара як основа вейвлету, вибирається 2-рівнева шкала розкладання, виконується вейвлет-перетворення, стискається реконструйоване зображення, встановлюючи порогове значення, і обчислюється співвідношення розміру файлу та значення

PSNR до та після стиснення. Функція вейвлет-реконструкції використовує матрицю коефіцієнтів, інформацію про розмірність і тип бази вейвлетів, отримані шляхом вейвлет-декомпозиції, для виконання вейвлет-реконструкції. Вибирається метод встановлення глобального порогу, і високочастотні вейвлет-коефіцієнти фільтруються пороговими значеннями для досягнення стиснення перед операцією реконструкції, а потім стиснене зображення отримується за допомогою вейвлет-реконструкції.

Після того, як зображення пройшло вейвлет-перетворення із заданим масштабом, більша частина енергії зосереджена в дробовій частині коефіцієнтів вейвлет-розкладання. Коефіцієнти інших частин встановлюються постійними шляхом встановлення порогу, і лише кілька коефіцієнтів розкладання зберігаються для представлення всього зображення. Результати експерименту показують, що простір для зберігання стисненого зображення значно зекономлено, а стиснене зображення не змінюється. візуально значно відрізняється від вихідного зображення.

Структурна схема розробленої системи зображена на рисунку 3.1. На ній показано структуру стиску та розтиснення з використанням вейвлет-перетворень.

Зберігання і передача зображень при цифровому представлені у вигляді матриці пікселів потребує обробки великих об'ємів даних. Проте безпосереднє представлення зображення у нестиснутому вигляді є неефективним унаслідок значної корельованості елементів матриці, а варіант незалежного кодування пікселів породжує надмірні коди. Графічне зображення, це – двовимірний сигнал s, двовимірна матриця кінцевого розміру. Застосовний до кожного рядка матриці один крок одновимірного вейвлет-перетворення. В результаті вийде дві матриці, рядки яких містять низькочастотні (НЧ) і високочастотні (ВЧ) складові рядків початкової матриці. До кожного стовпця обох матриць також застосовний крок одновимірного перетворення. В результаті виходить чотири матриці, які можна позначити як НЧ/НЧ, НЧ/ВЧ, ВЧ/НЧ і ВЧ/ВЧ. Перша є НЧ складовою початкового сигналу, інші три містять деталізуючу інформацію. Таким чином, на

кожному кроці двовимірного перетворення – присутня композиція одновимірних перетворень.

Нехай є одновимірний дискретний сигнал $s = \{s_j\}_{j\in Z}$. Кожній парі елементів з індексами 2j і 2j+1, $j \in Z$, поставимо у відповідність два значення:

$$\upsilon_j = \frac{s_{2j} + s_{2j+1}}{2}, \tag{3.1}$$

$$\omega_j = \frac{s_{2j} - s_{2j+1}}{2} \tag{3.2}$$

Ці значення формують два нові сигнали $v = \{\upsilon_j\}_{j\in Z}$ і $w = \{\omega_j\}_{j\in Z}$, один з яких є огрубленою версією початкового сигналу (кожній парі елементів s відповідає їх середнє арифметичне), а інший містить інформацію (називатимемо її деталізуючою), необхідну для відновлення початкового сигналу. Дійсно:

$$\begin{aligned} s_{2j} &= \upsilon_i + \omega_j, \\ s_{2j+1} &= \upsilon_i - \omega_j; \\ j &\in Z. \end{aligned} \tag{3.3}$$

До сигналу v можна застосувати аналогічну операцію і так само одержати два сигнали, один з яких є огрубленою версією v, а інший містить деталізуючу інформацію, необхідну для відновлення v.

Можна поставити у відповідність сигналу s довільний рівень дозволу i_1. Тоді рекурсивні формули для обчислення елементів сигналів для будь-якого дозволу $i_0 < i_1$:

$$\begin{aligned} \upsilon_j^{(i)} &= \frac{\upsilon_{2j}^{(i+1)} + \upsilon_{2j+1}^{(i+1)}}{2}, \\ \omega_j^{(i)} &= \frac{\upsilon_{2j}^{(i+1)} - \upsilon_{2j+1}^{(i+1)}}{2}, \\ i &= i_1 - 1,\ i_1 - 2, \ldots < i_0,\ j \in Z; \\ \upsilon_j^{(i_1)} &= s_j,\ j \in Z. \end{aligned} \tag{3.4}$$

Відновлення сигналу виконується за формулами:

$$\upsilon_{2j}^{(i+1)} = \upsilon_j^{(i)} + \omega_j^{(i)},\ \upsilon_{2j+1}^{(i+1)} = \upsilon_j^{(i)} - \omega_j^{(i)},$$

$$j \in Z,\ i = i_0, i_0 + 1, ..., i_1 - 1. \tag{3.5}$$

З проведеного аналізу можна зробити висновок, що ВП можна застосовувати для обробки цифрових даних. Але питання застосування дискретного ВП, особливо для стиснення зображень, є недостатньо вивчені. З метою подальшого розвитку й застосування дискретного ВП для обробки цифрових зображень, у запропонованій роботі розроблено алгоритм стиснення цифрових зображень з використанням ВП.

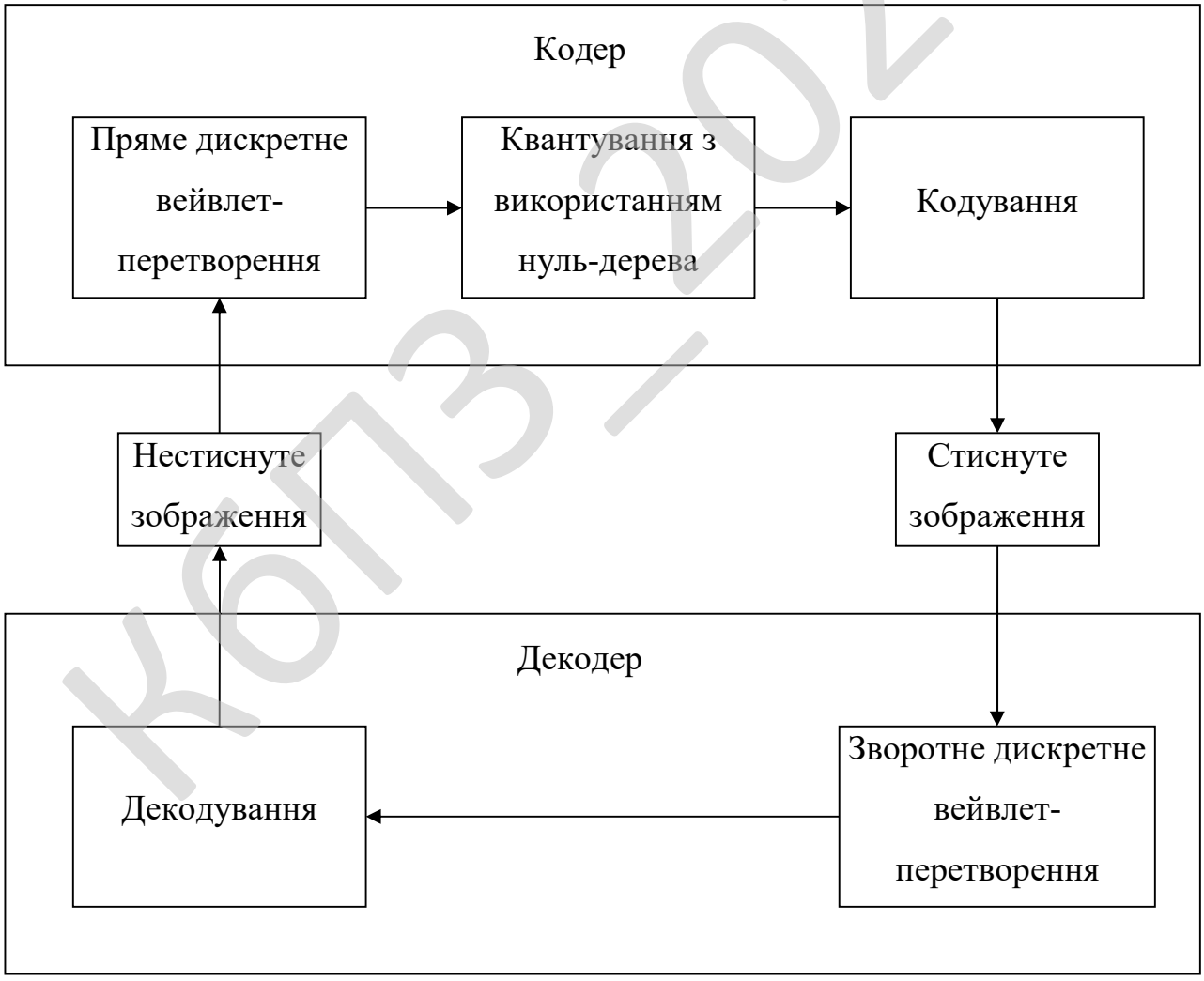

Рисунок 3.1 – Структурна схема системи

На практиці для стиснення зображень використовують різні дискретні перетворення. Найчастіше вихідне дискретне зображення розбивається для обробки на блоки невеликої розмірності та кожний фрагмент обробляється незалежно за допомогою перетворення при стиснені та зворотного перетворення при відновлені.

Схема розробленого алгоритму стиснення зображень з використанням ВП наведена у верхній частині структурної схеми.

Для відновлення зображення необхідно здійснити зворотні перетворення, схема відновлення зображення наведена у нижній частині структурної схеми.

Розглянемо алгоритм стиснення, частину першу. Кодер складається з трьох основних частин: прямого дискретного ВП, процедури квантування з використанням нуль-дерева та кодування. Розглянемо кожну складову вейвлет-кодера окремо. Пряме дискретне ВП. Для здійснення прямого дискретного ВП необхідно здійснити вибір базисної вейвлет-функції (ВФ), вибір якої є досить неоднозначною задачею. При виборі ВФ необхідно враховувати її властивості: наявності швидкого ВП, кількість нульових моментів, ортогональність. Перерахованими властивостями володіють ортогональні вейвлети (Добеши, Симлети, Койфлети) та В- сплайнові біортогональні вейвлети.

Квантування з використанням нуль-дерева. У алгоритмі застосовуються деревовидна структура даних для опису вейвлет-коефіцієнтів. Така структура виходить в результаті застосування двоканального роздільного ВП. Вузли дерева відповідають вейвлет-коефіцієнтам масштабу розкладу. Кожен вузол має чотири складових, відповідних вейвлет-коефіцієнтам наступного рівня. Квантування нуль-дерева засноване на принципі, що якщо коефіцієнт малий, то і всі його складові теж малі. Це пояснюється тим, що значущі коефіцієнти виникають поблизу контурів і текстур. Дерево або піддерево, яке містить тільки незначущі коефіцієнти, називається нуль-деревом. Спочатку кожен вузол квантується. Якщо значення квантованого вузла менше деякого порогу, піддерево, що починається з цього вузла, оголошується нуль-деревом, і його складові ігноруються. Ці складові

будуть відновлені декодером як нулі. Якщо вузол не має складових, починає оброблятися наступний кореневий вузол. Особливу ефективність цьому алгоритму додає сумісне кодування нулів за допомогою кодера.

Кодування. У даному алгоритмі використовується векторне кодування. В цьому випадку складається кодова книга, в якій зберігаються комбінації двохмірних блоків коефіцієнтів. Кожному блоку приписано його код, який отримано за алгоритмом Хаффмана. Кодування складається з находження для кожного кодованого блока з кодової книги середньоквадратичної різниці. У вихідний потік записуються коди блоків.

Для відновлення зображення рис. 3.1 проводиться декодування а потім здійснюється зворотне дискретне ВП.

Критерієм якості обробки зображень обрано середньоквадратичне відхилення відновленого зображення.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. З рисунку видно, що розроблена система складається з наступних частин:

1. Інтерфейс користувача програмного забезпечення стиснення зображень за допомогою вейвлет-перетворень.
2. Блок відкриття різних типів зображень.
3. Блок оптимізації, стиску й збереження зображень.
4. Вікно перегляду результатів стиску.
5. Вікно відображень розмірів файлів (вихідного й стислого).
6. Вікно роботи з метаданими зображення.
7. Набір стандартних інструментів (поворот, масштабування, відбиття зображення як горизонтально, так і вертикально).
8. Блок змін яскравості, контрастності, гами, і інвертування.
9. Блок налаштування файлів JPEG.

10. Блок налаштування файлів GIF.

11. Блок налаштування файлів PNG.

12. Блок налаштування маски зображення.

Рисунок 3.2 – Функціональна схема системи

Основні характеристики:

– програма може відкривати велику кількість різних типів зображень, таких як: jpg, jpeg, jpe, jif, png, gif, bmp, tif, tiff, psd, ico, tga, targa, mng, jng, j2k, j2c, jp2, pcd, pcx, wpa, wbmp, wbm, xbm, xpm, dds, g3, koa, iff, lbm, pbm, pgm, ppm, ras, cut, sgi, pct, pict, pic;

– за допомогою програми можна оптимізувати й згодом зберегти вихідне зображення у форматах JPEG, GIF і PNG, причому для цього не прийдеться мати специфічні знання, інтерфейс програми настільки простий що розібратися зможе кожний;

– вікно програми розділене на два вікна: вихідне зображення й стиснуте, багато налаштувань застосовуються в режимі реального часу, без додаткового натискання клавіш – автоматичний перегляд результатів;

– можна виставити заданий розмір файлу, для стиснутого зображення;

– на екрані відразу відображається розмір підсумкового файлу й вихідного;

– у програмі можна працювати як з одним файлом, так і з декількома одночасно;

– є можливість роботи із прозорістю;

– можна поміняти метадані зображення (коментарі, IPTC, Adobe XMP, EXIF профайл, ICC профайл), не підтримувані метадані будуть видалені;

– можлива передача метаданих між зображеннями (але це в тому випадку якщо кінцевий файл підтримує такі типи метаданих);

– зі стандартних інструментів доступні: поворот, масштабування, і їсти можливість відбити зображення як горизонтально так і вертикально;

– можна так само можливо перемінити яскравість, контрастність, гаму, і інвертувати;

– у реальному часі є можливість зменшити кількість кольорів PNG і GIF, для того що б розмір файлу став менший;

– можлива зміна розміру зображення за допомогою відомих фільтрів таких як: Lanczos3, Catmull Rom, Bicubic, і ін.;

– у доповненнях можна знайти підтримку зовнішніх оптимізаторів зображення PNG (optiPNG, PNGOut).

Для файлів JPEG доступні наступні налаштування:

− якість стиску;

− розширені можливості вибору насиченості кольору (відсутній, низький 4:2:2, середній 4:2:0, високий 4:1:1);

− можна зберегти зображення як відтінки сірого 8-біт;

− стандартна оптимізація (оптимальні таблиці Хаффмана) або прогресивне кодування.

Підтримується на вибір наступна глибина кольору: 24 біта, 8 біт відтінки сірого.

Налаштування для файлів GIF:

1. Зміна кольору:

− зміна кольору від 256 до 2 кольорів (без згладжування) з використанням Xiaolin Wu або NeuQuant neural net;

− палітра 8 біт відтінки сірого;

− floyd-steinberg згладжування 1 біт монохромний.

2. Збереження черезрядкового.

На виході підтримується глибина: 4, 8 біт, 8 біт градацій сірого, 1 біт монохромний.

Налаштування для файлів PNG:

1. Підтримуються вихідні кольори (24 біт RGB, 32 біт RGBA).

2. Зміна кольору:

− зміна кольору від 256 до 2 кольорів (без згладжування) з використанням Xiaolin Wu або NeuQuant neural net;

− палітра 8 біт відтінки сірого;

− floyd-steinberg згладжування 1 біт монохромний.

3. Збереження черезрядкового.

4. Домогтися максимальних налаштувань стиску можна шляхом використання інтеграції з популярними оптимізаторами PNG (optiPNG, PNGOut):

− додати / видалити / змінити зовнішній інструмент оптимізації PNG.

На виході підтримується глибина: 4, 8, 24, 32 бітний колір, 8 біт градацій сірого, 1 біт монохромний.

Налаштування для метаданих

Можна зберегти або видалити наступні метадані:

− коментарі;

− Adobe XMP інформація;

− IPTC інформація;

− профіль EXIF (у тому числі GPS і творець замітки);

− профілі кольору ICC.

Невідомі або не підтримувані метадані автоматично видаляються.

Налаштування маски

У програмі можна вибрати кілька варіантів прозорості:

− Зберігати прозорість (використання порога для переходу від альфа до індексованої прозорості).

− Суміш із чистим тлом (можна вибрати колір для змішування прозорості у фоновому режимі (альфа состав)).

− Можна зробити не прозорим (можливість видалення прозорості інформації, роблячи всі непрозорим).

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Першим процесом, який завантажується у системі, є процес виведення головного вікна програми. Він взаємодіє з наступними процесами:

– Процес виведення параметрів вейвлет-стиснення.

– Процес відкриття стиснутого графічного файлу.

– Процес відкриття нестиснутого графічного файлу.

Рисунок 3.3 – Діаграма взаємодії процесів

Процес виведення параметрів вейвлет-стиснення взаємодіє з наступними процесами:

– Процес введення коефіцієнту вейвлет-стиснення.

– Процес введення підрозділу глибин.

– Процес введення коефіцієнтів квантування.

Процес відкриття стиснутого графічного файлу взаємодіє з процесом розпакування файлу для виведення на екран, який, у свою чергу, взаємодіє з процесом виведення на екран стиснутого файлу.

Процес відкриття нестиснутого графічного файлу взаємодіє з наступними процесами:

– Процес виведення на екран нестиснутого графічного файлу, який, у свою чергу, взаємодіє з процесом виведення розміру нестиснутого файлу.

– Процес вейвлет-стиснення файлу з вказаними параметрами, який взаємодіє з наступними процесами:

а) процес виведення розміру стиснутого файлу;

б) процес збереження стиснутого файлу;

в) процес виведення на екран стиснутого файлу.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми, блок-схема якого зображена на рисунку 4.1.

Після запуску програми відбувається виведення основного вікна програми.

Далі пропонується відкрити зображення, після чого виводиться зображення на екран із зазначенням розміру файлу із зображенням.

Основний режим роботи програми полягає у виконанні процедур стиснення зображень та декомпресії.

При встановленні користувачем режиму стиснення зображення, спочатку вибираються параметри вейвлет-стиснення. Далі виконується підпрограма стиснення зображень за допомогою вейвлет-перетворень. Потім на екран виводиться зображення після стиснення із зазначенням розміру стиснутого зображення.

При встановленні користувачем режиму декомпресії зображення, виконується підпрограма декомпресії за допомогою вейвлет-перетворення. Далі виконується виведення на екран зображення після декомпресії.

Після збереження вказаного зображення у файл користувачеві пропонується вихід із програми.

Вейвлет перетворення застосовується в різних галузях прикладної науки, таких як обробка сигналів і зображень. Найбільш широке застосування дискретне вейвлет-перетворення використовується в кодуванні сигналів, де властивості перетворення використовуються для зменшення надмірності в представленні дискретних сигналів, часто – як перший етап в компресії даних.

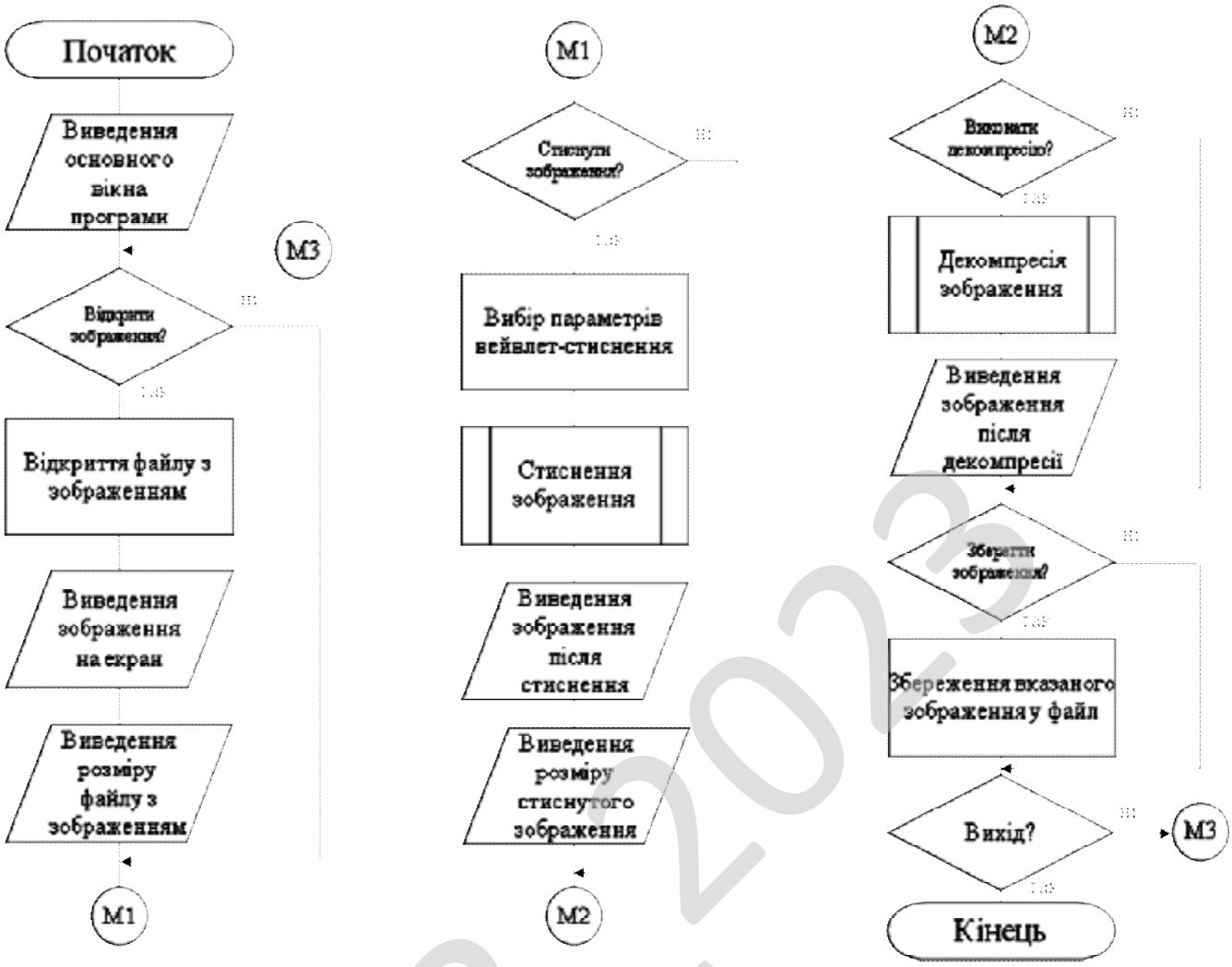

Рисунок 4.1 – Блок-схема основної програми

Алгоритм, який використовується в роботі для стиснення зображень, заснований на отриманні за допомогою двохетапного дискретного вейвлет-преобразования чотирьох зображень:

а) LL – і в горизонтальному, і у вертикальному напрямі низькочастотні відліки;

б) LH – в горизонтальному низькочастотні, у вертикальному високочастотні;

в) HL – в горизонтальному високочастотні, у вертикальному низькочастотні;

г) HH – в обох напрямах високочастотні.

Потім відліки цих зображень подаються як вхідні дані для алгоритму стиснення GZIP у вигляді одновимірного вектора розмірністю чотири.

Суть методу полягає в розкладанні сигналу по базисних функціях локалізованим як в просторі, так і в часі. Це дозволяє враховувати нестаціонарну поведінку коефіцієнтів вейвлет-перетворення. Крім того, масштабованість базисних функцій дозволяє аналізувати сигнал зображення з різним ступенем деталізації, що знаходить своє застосування при стисканні зображень.

Стискання сигналів здійснюється через набори банку фільтрів із заданими властивостями здійснюючи ітераційно декомпозицію зображення.

При цьому фільтри підбираються так, щоб забезпечити точне відновлення початкового сигналу по вейвлет-коефіцієнтам. Ця умова накладає певні обмеження на характеристики використовуваних фільтрів. Так, наприклад, фільтри аналізу і синтезу повністю або частково ортогональні один одному.

Розглянемо детальніше роботу підпрограми стиснення зображень за допомогою вейвлет-перетворень.

Після початкового вейвлет-перетворення здійснюється визначення вейвлет-коефіцієнтів.

Наступним кроком встановлюється поріг T, рівний найбільшій степені двійки, що менше максимального коефіцієнта.

Далі береться значення низькочастотного коефіцієнта LL.

Доки T>=1 виконується наступна послідовність операцій:

– Символьний масив ініціалізується нулями.

– Перехід знизу вгору для LH.

– Перехід згори вниз для LH.

– Перехід знизу вгору для HH.

– Перехід згори вниз для HH.

– Перехід згори вниз для HL.

– Зменшення порогу T в 2 рази.

Після цього відбувається перехід до нового розбиття частин зображення на високо- та низькочастотні області.

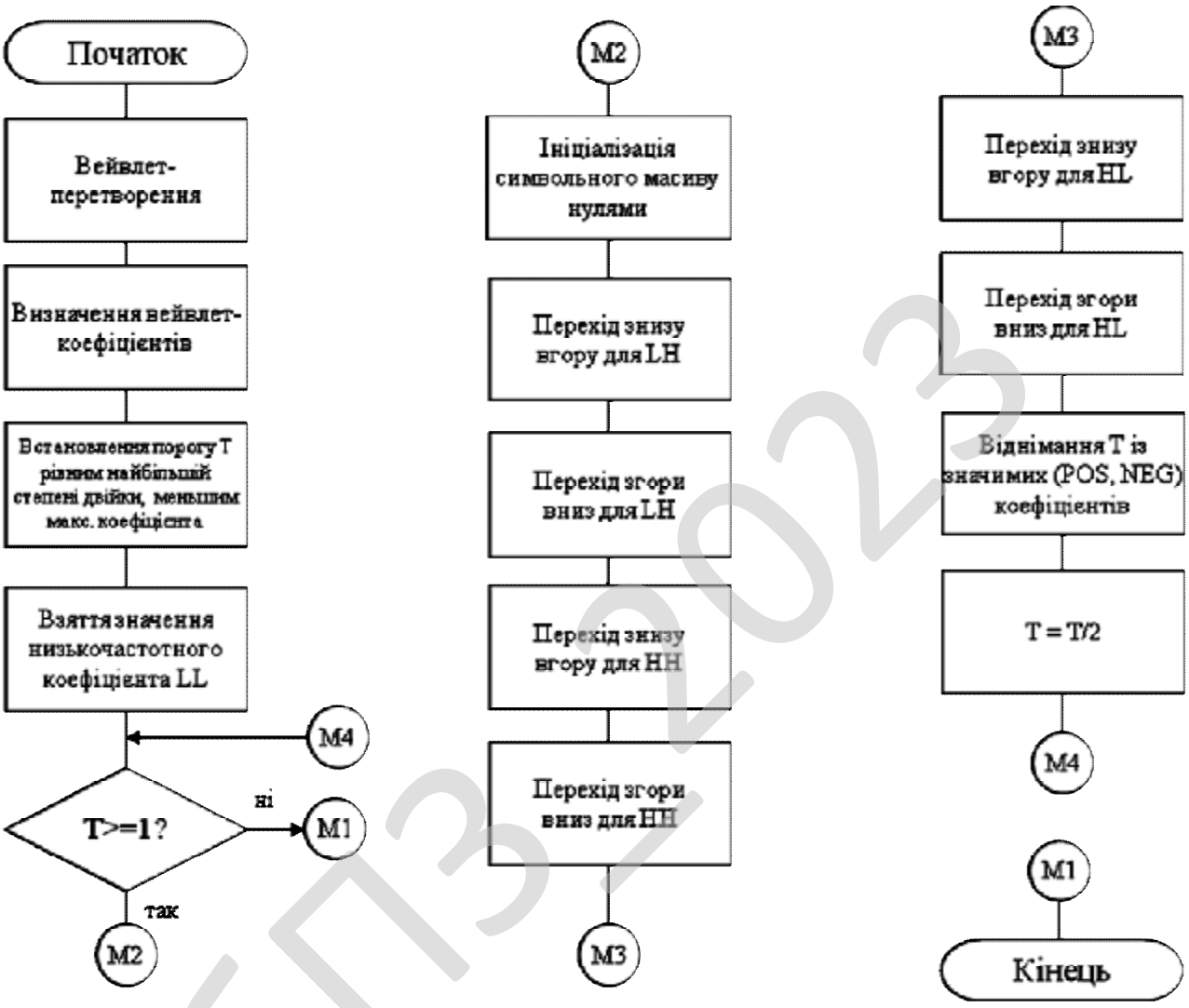

Рисунок 4.2 – Блок-схема підпрограми стиснення зображень за допомогою вейвлет-перетворень

Попередньо зображення слід перетворити із формату RGB в YCrCb. Дана операція здійснюється за допомогою наступного методу:

```
        private double[, ,] YCrCbEncode(byte[, ,] BytesRGB, int c, int c, double
Ydiv, double Udiv, double Vdiv, int o, int o)
        {
            double vr, vg, vb;
```

```
            double kr = 0.299, kg = 0.587, kb = 0.114, kr1 = -0.1687, kg1 = 0.3313,
kb1 = 0.5, kr2 = 0.5, kg2 = 0.4187, kb2 = 0.0813;
            Ydiv = Ydiv / 100f;
            Udiv = Udiv / 100f;
            Vdiv = Vdiv / 100f;
            double[, ,] YCrCb = new double[3, c, c];
            for (int j = 0; j < o; j++)
            {
                for (int i = 0; i < o; i++)
                {
                    vb = (double)BytesRGB[0, i, j];
                    vg = (double)BytesRGB[1, i, j];
                    vr = (double)BytesRGB[2, i, j];
                    YCrCb[2, i, j] = (kr * vr + kg * vg + kb * vb) * Ydiv;
                    YCrCb[1, i, j] = (kr1 * vr - kg1 * vg + kb1 * vb + 128) * Udiv;
                    YCrCb[0, i, j] = (kr2 * vr - kg2 * vg - kb2 * vb + 128) * Udiv;
                }
            }
            return YCrCb;
        }
```

Кожен канал кодується окремо із різним ступенем стиснення.

Для роботи з графікою необхідно обробляти двовимірні масиви даних. Кратномасштабний аналіз повинен будуватися в просторі функцій два змінних. Існує декілька способів узагальнити викладені раніше конструкції на функції декілька змінних (ми говоритимемо тільки про функції два змінних).

Найпростіший і широко поширений шлях – тензорний добуток одновимірних КМА. Як двовимірна скейлінг-функція береться

$$\Phi(t,s) = \varphi(t)\varphi(s).$$

Замість одного вейвлету виникає три:

$$\Psi_{LH}(t,s) = \varphi(t)\psi(s), \Psi_{HL}(t,s) = \psi(t)\varphi(s), \Psi_{HH}(t,s) = \psi(t)\psi(s).$$

(L означає низьку частоту, H – високу частоту). Простори породжуються зсувами скейлінг – функції на одному і тому ж масштабі:

$$V_j = \left\{2^j \Phi(2^j t - k, 2^j s - m)\right\},$$

простори деталей мають вигляд:

$$W^{LH}{}_j = \left\{2^j \Psi_{LH}(2^j t-k, 2^j s-m)\right\}, W^{HL}{}_j = \left\{2^j \Psi_{HL}(2^j t-k, 2^j s-m)\right\},$$
$$W^{HH}{}_j = \left\{2^j \Psi_{HH}(2^j t-k, 2^j s-m)\right\}$$

Відповідні проекції знаходяться застосуванням фільтрів:

$$x \mapsto \left\{H_t \otimes H_s(x), H_t \otimes G_s(x), G_t \otimes H_s(x), G_t \otimes G_s(x)\right\},$$

де x – двовимірний сигнал. Класична схема Малла передбачає рекурсивне застосування тієї ж процедури до низькочастотної складової. Коефіцієнти обираються згідно правила вказаного на рисунку 4.3:

LL	**HL**
LH	**HH**

Рисунок 4.3 – Правило розбиття на низько-(L) та високочастотні(H) квадранти по вертикалі та горизонталі

Наприклад, *LH* означає, що в цьому квадранті стоїть результат застосування фільтру низьких частот до стовпців, високих частот – до строчок початкової матриці, і проріджування удвічі по кожному напряму. Коефіцієнти більшої амплітуди вказують на різкі перепади яскравості. Такі перепади є найбільш інформативними при побіжному перегляді будь-якого зображення. Вейвлет-представлення дозволяє їх локалізувати шляхом послідовного уточнення, починаючи з більш крупних масштабів. Крім того, коефіцієнти проекції на різні простори деталей відповідають за перепади яскравості різної орієнтації: наприклад, якщо фільтр високих частот застосовувався до рядків, то у відповідному квадранті яскравіше виділені *вертикальні* перепади.

Найпростіший підхід до стиснення зображень за допомогою вейвлет-перетворення полягає в наступному:

– Виконати вейвлет-перетворення.

– Упорядкувати коефіцієнти .

– Відкинути “хвіст” впорядкованого масиву, енергія якого дорівнює допустимій (за умовами завдання) величині.

– Запам'ятати збережені коефіцієнти і їх положення в масиві початкових коефіцієнтів.

– При відновленні замінювати відкинуті коефіцієнти нулями.

Ця ідея в тій або іншій формі присутня у всіх методах вейвлетного стискування. Дану процедуру можна, наприклад, застосовувати роздільно до кожного з квадрантів, отриманих при розкладанні по оптимально вибраних вейвлет-пакетам.

Вейвлет-перетворення виконується за допомогою наступної функції:

```
private double[, ,] WaveletePack(double[, ,] ImgArray, int Component, int c, int c, int dwDevider, int dwTop, int dwStep)
        {
            short Value;
            int cw2 = c / 2;
            int c2 = c / 2;
            double dbDiv = 1f / dwDevider;
            ImgArray = Wv(ImgArray, c, c, Component, WV_TOP_TO_BOTTOM);
            ImgArray = Wv(ImgArray, c, c, Component, WV_LEFT_TO_RIGHT);
            // квантування
            for (int j = 0; j < c; j++)
            {
                for (int i = 0; i < c; i++)
                {
                    if ((i >= cw2) || (j >= c2))
                    {
                        Value = (short)Math.Round(ImgArray[Component, i, j]);
                        if (Value != 0)
                        {
                            int value2 = Value;
                            if (value2 < 0) { value2 = -value2; }
                            if (value2 < dwTop)
                            {
                                ImgArray[Component, i, j] = 0;
                            }
```

```
                        else
                        {
                            ImgArray[Component, i, j] = Value * dbDiv;
                        }
                    }
                }
            }
        }
        return ImgArray;
    }
```

Процедура швидкого ліфтингу дискретного біортогонального вейвлету, що використовується в вейвлет-перетворенні:

```
    private double[, ,] Wv(double[, ,] ImgArray, int n, int dwCh, int Component,
int Side)
    {
        double a;
        int i, j, n2 = n / 2;
        double[] xWavelet = new double[n];
        double[] tempbank = new double[n];
        for (int dwPos = 0; dwPos < dwCh; dwPos++)
        {
            if (Side == WV_LEFT_TO_RIGHT)
            {
                for (j = 0; j < n; j++) {
                    xWavelet[j] = ImgArray[Component, dwPos, j];
                }
            }
            else if (Side == WV_TOP_TO_BOTTOM)
            {
                for (i = 0; i < n; i++) {
                    xWavelet[i] = ImgArray[Component, i, dwPos];
                }
            }
            // Прогноз 1
            a = -1.586134342f;
            for (i = 1; i < n - 1; i += 2) {
                xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
            }
            xWavelet[n - 1] += 2 * a * xWavelet[n - 2];
            // Оновлення 1
            a = -0.05298011854f;
```

```
            for (i = 2; i < n; i += 2) {
                xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
            }
            xWavelet[0] += 2 * a * xWavelet[1];
            // Прогноз 2
            a = 0.8829110762f;
            for (i = 1; i < n - 1; i += 2) {
                xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
            }
            xWavelet[n - 1] += 2 * a * xWavelet[n - 2];
            // Оновлення 2
            a = 0.4435068522f;
            for (i = 2; i < n; i += 2) {
                xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
            }
            xWavelet[0] += 2 * a * xWavelet[1];
            // масштаб
            a = 1f / 1.149604398f;
            j = 0;
            // множимо непарні на коефіцієнт "a"
            // ділимо парні на коефіцієнт "a"
            if (Side == WV_LEFT_TO_RIGHT)
            {
                for (i = 0; i < n2; i++) {
                    ImgArray[Component, dwPos, i] = xWavelet[j++] / a;
                    ImgArray[Component, dwPos, n2 + i] = xWavelet[j++] * a;
                }
            }
            else if (Side == WV_TOP_TO_BOTTOM)
            {
                for (i = 0; i < n2; i++) {
                    ImgArray[Component, i, dwPos] = xWavelet[j++] / a;
                    ImgArray[Component, n2 + i, dwPos] = xWavelet[j++] * a;
                }
            }
        }
        return ImgArray;
    }
```

В продвинутих методах кодування відразу йде на рівні бітів. Один з них ми зараз коротко опишемо. Він є модифікацією методу *вкладеного нуль-дерева (embedded zero-tree)*.

– Коефіцієнти вейвлет-перетворення квантуються і записуються цілими числами.

– Будується впорядкована таблиця коефіцієнтів: спочатку йдуть ті, біля яких в старшому двійковому розряді 1, потім – ті, біля яких в старшому розряді 0, але в наступному – 1, і так далі. В бітовому представлення схема побудови таблиці коефіцієнтів представлена на рисунку 4.4.

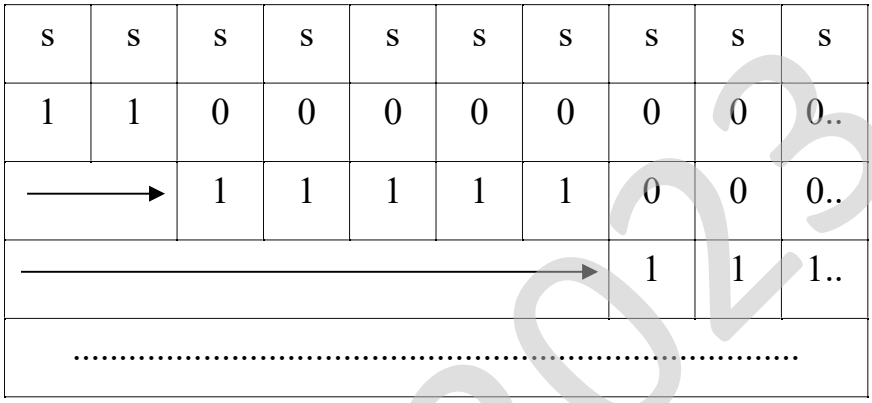

Рисунок 4.4 – Побудова таблиці коефіцієнтів

У першому рядку стоять знаки коефіцієнтів, в другій – старші біти, і так далі Ясно, що для відтворення початкового зображення досить запам'ятати (передати) тільки ті біти, які стоять в клітках, помічених стрілками, а також довжину стрілок і положення самих коефіцієнтів в початковому двовимірному масиві коефіцієнтів. При передачі бітів в такому порядку спочатку передається найістотніша інформація (старші біти найбільших коефіцієнтів), потім менш істотна, і так далі. Цей процес можна обірвати, дійшовши до тієї частки таблиці, де стоять «маленькі» коефіцієнти. Проблема в тому, як компактно передати таблицю положень збережених коефіцієнтів. Відмітимо, що повністю упорядковувати масив коефіцієнтів (по абсолютній величині) не треба, досить знайти розбиття на групи, показані в таблиці, тобто на такі групи, що для фіксованого n

$$2^n \le \left|c_{ij}\right| < 2^{n+1}.$$

Нехай вся множина коефіцієнтів розбита на деякі підмножини . Фіксуємо n, і починаємо по черзі переглядати T_m на наявність *значущих* коефіцієнтів – тобто, що задовольняють останнє рівняння. Якщо в T_m є значущі коефіцієнти, ця множина розбивається на деякі підмножини, і вони знову перевіряються на значущість, і так далі. Будувати розбиття намагаються так так, щоб значуща множина складалася (майже завжди) з єдиного елементу, а незначущі були великими.

Саме тут використовується специфіка вейвлет-аналізу. У розкладах такого типа масив коефіцієнтів розпадеться на набір піддерев. Розглянемо структуру дерев, на які розпадається множина вейвлет-коефіцієнтів. Приклад такого розбиття зображено на рисунку 4.5.

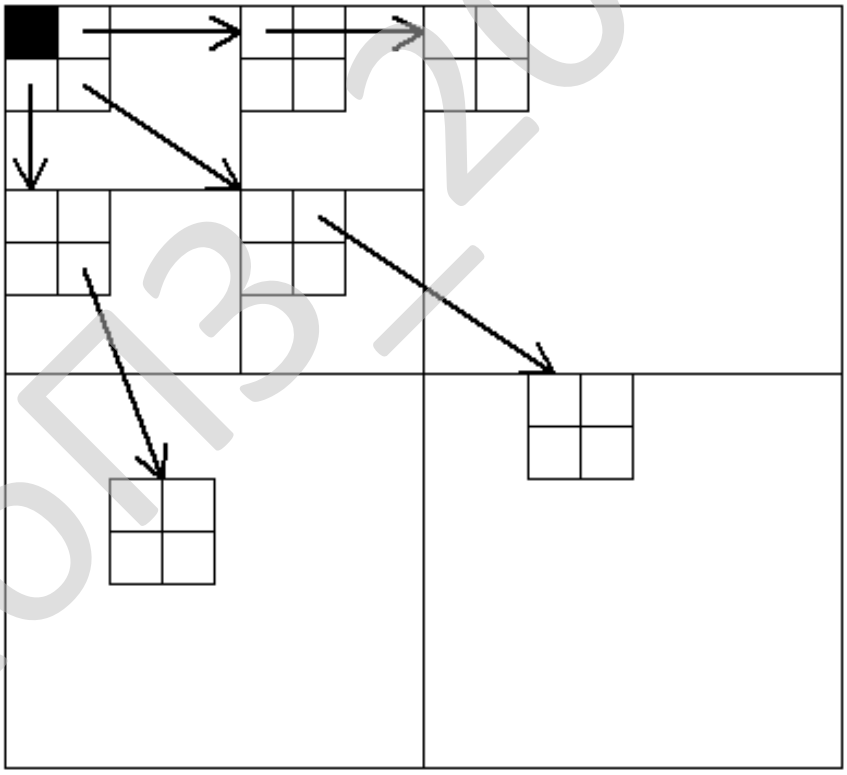

Рисунок 4.5 – Структура дерев, на які розпадається множина вейвлет-коефіцієнтів

Верхній лівий квадрант розпадається на четвірки коефіцієнтів. У кожній четвірці три коефіцієнти (окрім закрашеного чорним) мають «нащадків» – четвірки коефіцієнтів на нижніх рівнях; *кожен* з коефіцієнтів в цих четвірках має своїх нащадків, і так далі. Вузлу *(i,j)* відповідають нащадки з координатами *(2i,2j) (2i,2j+1) (2i+1,2j) (2i+1,2j+1)*. Як початкове розбиття береться саме цей набір піддерев. Багато хто з них дійсно складається тільки з неістотних коефіцієнтів, оскільки малі значення на верхніх рівнях майже завжди відповідають малим значенням на нижніх рівнях (якщо рухатися по напряму стрілок на мал. 5). Не заглиблюючись в подальші подробиці, відзначимо ще одну особливість цього красивого алгоритму: передається (запам'ятовується) не таблиця положень коефіцієнтів, а ключові кроки процесу сортування, що істотно компактніше (при декодуванні цей процес фактично відтворюється в зворотному ладі). Оскільки зазвичай багато піддерев виявляються “нульовими”, кроків сортування буде не дуже багато.

На рисунку 4.6 зображено блок-схему підпрограми декомпресії зображень після вейвлет-перетворень.

На початку роботи цієї підпрограми виконується читання даних заголовка, а саме: тип вейвлета, кількість рядків, низькочастотне значення (НН) та бітові площини.

На наступному кроці обчислюється поріг $T = 2^N$.

Після цього циклічно, доки T>=1, виконується наступна послідовність операцій:

– Символьний масив ініціалізується нулями.

– Зчитуються символи для піддіапазону LH згори вниз.

– Зчитуються символи для піддіапазону HH згори вниз.

– Зчитуються символи для піддіапазону HL згори вниз.

– Отримані величини додаються до значимих (POS, NEG) коефіцієнтів.

– Поріг T зменшується у 2 рази.

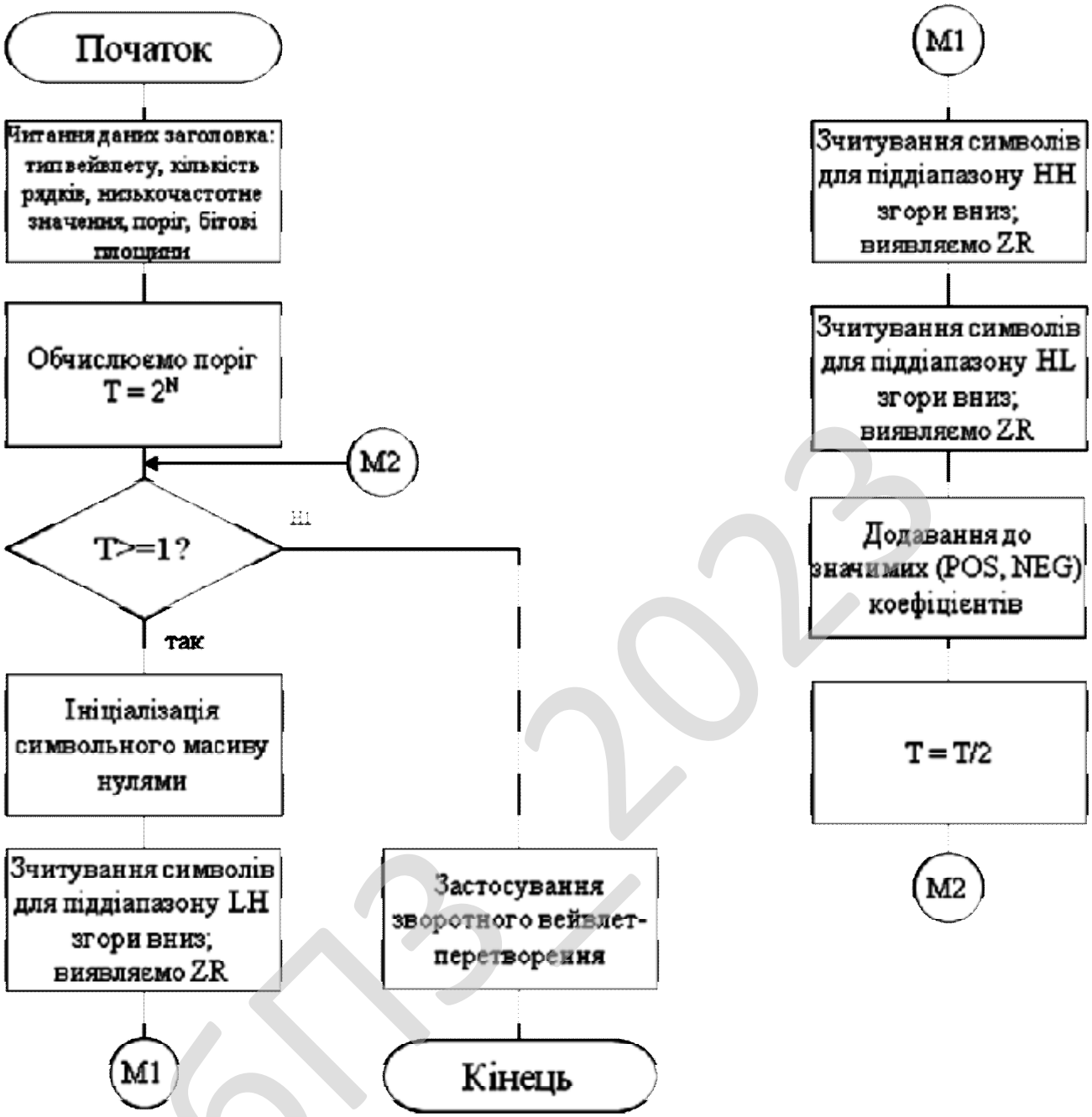

Рисунок 4.6 – Блок-схема підпрограми декомпресії зображень після вейвлет-перетворень

Після цього відбувається зворотне вейвлет-перетворення і програма завершує свою роботу.

Розгортання вейвлету відбувається за допомогою наступного методу:

```
private void WaveleteUnPack(double[,,] ImgArray, int Component, int c, int c, int dwDevider)
```

```
        {
            int cw2 = c / 2, ch2 = c / 2;
            double dbDiv = 1f / dwDevider;
            // деквантування значень
            for(int i = 0; i < c; i++)
            {
                for(int j = 0; j < c; j++)
                {
                    if ((i >= cw2) || (j >= ch2))
                    {
                        if (ImgArray[Component, i, j] != 0)
                        {
                            ImgArray[Component, i, j] /= dbDiv;
                        }
                    }
                }
            }
            for(int i = 0; i < c; i++)
            {
                reWv(ref ImgArray, c, Component, i, WV_LEFT_TO_RIGHT);
            }
            for(int j = 0; j < c; j++)
            {
                reWv(ref ImgArray, c, Component, j, WV_TOP_TO_BOTTOM);
            }
        }
```

Процедура зворотного швидкого ліфтингу дискретного біортогонального вейвлету, що застосовується в розгортанні вейвлету:

```
        private void reWv(ref double[, ,] shorts, int n, int z, int dwPos, int
Side)
        {
            double a;
            double[] xWavelet = new double[n];
            double[] tempbank = new double[n];
            if(Side == WV_LEFT_TO_RIGHT)
            {
                for(int j = 0; j < n; j++)
                {
                    xWavelet[j] = shorts[z, dwPos, j];
                }
            }
```

```
            else if (Side == WV_TOP_TO_BOTTOM)
            {
                for(int  i = 0; i < n; i++)
                {
                    xWavelet[i] = shorts[z, i, dwPos];
                }
            }
            for(int i = 0; i < n / 2; i++)
            {
                tempbank[i * 2] = xWavelet[i];
                tempbank[i * 2 + 1] = xWavelet[i + n / 2];
            }
            for(int i = 0; i < n; i++)
            {
                xWavelet[i] = tempbank[i];
            }
            // відмінити масштабування
            a = 1.149604398f;
            for(int  i = 0; i < n; i++)
            {
                if(i % 2 != 0)
                {
                    xWavelet[i] = xWavelet[i] * a;
                } else {
                    xWavelet[i] = xWavelet[i] / a;
                }
            }
            // Повернути оновлення 2
            a = -0.4435068522f;
            for (int i = 2; i < n; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[0] = xWavelet[0] + 2 * a * xWavelet[1];
            // повернути прогнозування 2
            a = -0.8829110762f;
            for (int i = 1; i < n - 1; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
```

```
            xWavelet[n - 1] = xWavelet[n - 1] + 2 * a * xWavelet[n - 2];
            // Повернути оновлення 1
            a = 0.05298011854f;
            for (int i = 2; i < n; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[0] = xWavelet[0] + 2 * a * xWavelet[1];
            // повернути прогнозування 1
            a = 1.586134342f;
            for (int i = 1; i < n - 1; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[n - 1] = xWavelet[n - 1] + 2 * a * xWavelet[n - 2];
            if(Side == WV_LEFT_TO_RIGHT)
            {
                for (int j = 0; j < n; j++)
                {
                    shorts[z, dwPos, j] = xWavelet[j];
                }
            }
            else if(Side == WV_TOP_TO_BOTTOM)
            {
                for(int i = 0; i < n; i++)
                {
                    shorts[z, i, dwPos] = xWavelet[i];
                }
            }
        }
```

Для отримання початкового зображення слід провести зворотне перетворення із YCrCb в RGB:

```
        private byte[] YCrCbDecode(double[, ,] yuv, int w, int h, double Ydiv,
double Udiv, double Vdiv)
        {
            byte[] bytes_flat = new byte[3 * w * h];
            double vr, vg, vb;
            double v, vCb, vCr;
            Ydiv = Ydiv / 100f;
```

```
            Udiv = Udiv / 100f;
            Vdiv = Vdiv / 100f;
            for(int j = 0; j < h; j++)
            {
                for (int i = 0; i < w ; i++)
                {
                    vCr = yuv[0, i, j] / Vdiv;
                    vCb = yuv[1, i, j] / Udiv;
                    v = yuv[2, i, j] / Ydiv;
                    vr = v + 1.402f * (vCr - 128f);
                    vg = v - 0.34414f * (vCb - 128f) - 0.71414f * (vCr - 128f);
                    vb = v + 1.722f * (vCb - 128f);
                    if (vr > 255) {vr = 255;}
                    if (vg > 255) {vg = 255;}
                    if (vb > 255) {vb = 255;}
                    if (vr < 0) {vr = 0;}
                    if (vg < 0) {vg = 0;}
                    if (vb < 0) {vb = 0;}
                    bytes_flat[j * w * 3 + i * 3 + 0] = (byte)vb;
                    bytes_flat[j * w * 3 + i * 3 + 1] = (byte)vg;
                    bytes_flat[j * w * 3 + i * 3 + 2] = (byte)vr;
                }
            }
            return bytes_flat;
        }
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Twofish, який є симетричним алгоритмом блочного шифрування з розміром блоку 128 біт і довжиною ключа до 256 біт. Число раундів 16. Розроблено групою фахівців на чолі з Брюсом Шнайером. Був одним з п'яти фіналістів другого етапу конкурсу AES. Алгоритм розроблений на основі алгоритмів Blowfish, SAFER і Square.

Відмінними особливостями алгоритму є використання попередньо обчислюваних та залежних від ключа S-box'ів і складна схема розгортки підключення шифрування. Половина n-бітного ключа шифрування

використовується як власне ключ шифрування, інша – для модифікації алгоритму (від неї залежать S-box'и).

Twofish розроблявся спеціально з урахуванням вимог та рекомендацій NIST для конкурсу AES [1]:

– 128-бітний блочний симетричний шифр.

– Довжина ключів 128, 192 і 256 біт.

– Відсутність слабких ключів.

– Ефективна програмна (в першу чергу на 32-бітних процесорах) та апаратна реалізація.

– Гнучкість (можливість використання додаткових довжин ключа, використання в поточному шифруванні, хеш-функціях і т.д.).

– Простота алгоритму – для можливості його ефективного аналізу.

Однак саме складність структури алгоритму і, відповідно, складність його аналізу на предмет слабких ключів або прихованих зв'язків, а також досить повільне час шифрування порівняно з Rijndael на більшості платформ, зіграло не на його користь.[2]

Алгоритм Twofish виник в результаті спроби модифіковані алгоритм Blowfish для 128-бітового вхідного блоку. Новий алгоритм повинен був бути легко реалізованим апаратно (у тому числі використовувати таблиці меншого розміру), мати досконалішу систему розширення ключа (key schedule) і мати однозначну функцію F.

В результаті, алгоритм був реалізований у вигляді змішаної мережі Фейстеля з чотирма гілками, які модифікують один одну з використанням криптоперетворень Адамара (Pseudo-Hadamar Transform, PHT).

Можливість ефективної реалізації на сучасних (для того часу) 32-бітних процесорах (а також в смарт-картах і подібних пристроях) – один з ключових принципів, яким керувалися розробники Twofish.

Наприклад, у функції F при обчисленні PHT і складання з частиною ключа K навмисно використовується додавання, замість традиційного xor. Це дає

можливість використовувати команду LEA сімейства процесорів Pentium, яка за один такт дозволяє обчислити перетворення Адамара. (Правда в такому випадку код доводиться компілювати під конкретне значення ключа).

Алгоритм Twofish не запатентований і може бути використаний ким завгодно без будь-якої плати або відрахувань. Він використовується в багатьох програмах шифрування, хоча і отримав менше поширення, ніж Blowfish.

Опис алгоритму

Twofish розбиває вхідний 128-бітний блок даних на чотири 32-бітних підблоки, над якими, після процедури вхідного відбілювання (input whitening), проводиться 16 раундів перетворень. Після останнього раунду виконується вихідна відбілювання (output whitening).

Відбілювання (whitening)

Відбілювання – це процедура xor'a даних з підключами перед першим раундом і після останнього раунду. Вперше ця техніка була використана в шифрі Khufu/Khare і, незалежно, Рональдом Рівестом (англ. Ron Rivest) в алгоритмі шифрування DESX. Joe Killian (NEC) і Phillip Rogaway (Каліфорнійський університет) показали, що відбілювання дійсно ускладнює завдання пошуку ключа повним перебором (exhaustive key-search) в DESX[3] . Розробники Twofish стверджують[4], що в Twofish відбілювання також істотно ускладнює завдання підбору ключа, оскільки криптоаналітика не може дізнатися, які дані потрапляють на вхід функції F першого раунду.

Тим не менш, виявилися і негативні сторони. Цікаве дослідження провели фахівці дослідницького центру компанії IBM.[5] Вони виконали реалізацію алгоритму Twofish для типової смарт-карти з CMOS-архітектурою і проаналізували можливість атаки за допомогою диференціального аналізу споживаної потужності (DPA – Differential Power Analysis). Атаці піддавалася саме процедура вхідного вибілювання, оскільки вона безпосередньо використовує xor підключів з вхідними даними. У результаті дослідники

показали, що можна повністю обчислити 128-бітовий ключ проаналізувавши всього 100 операцій шифрування довільних блоків.

Функція g

Функція g – основа алгоритму Twofish. На вхід функції подається 32-бітове число X, яке потім розбивається на чотири байти x0, x1, x2, x3. Кожен з вийшов байтів пропускається через свій S-box. (Слід зазначити, що S-box'и в алгоритмі не фіксовані, а залежать від ключа). Отримані 4 байти на виходах S-box'ов інтерпретуються як вектор з чотирма компонентами. Цей вектор множиться на фіксовану матрицю MDS (maximum distance separable) розміром 4x4, причому обчислення проводяться в скінченному полі по модулю непривідного многочлена

MDS матриця – це така матриця над кінцевим полем K, що якщо взяти її як матрицю лінійного перетворення з простору у простір , то будь-які два вектори з простору виду (x, f (x)) будуть мати як мінімум m+1 відмінностей в компонентах. Тобто набір векторів вигляду (x, f (x)) утворює код, що володіє властивістю максимального рознесення (maximum distance separable code). Таким кодом, наприклад, є код Ріда-Соломона.

В Twofish властивість максимальної рознесеність матриці MDS означає, що загальна кількість змінних байт вектора a і вектора не менше п'яти. Іншими словами, будь-яка зміна тільки одного байта в a призводить до зміни всіх чотирьох байтів в b.

Криптоперетворення Адамара (Pseudo-Hadamar Transform, PHT)

Криптоперетворення Адамара – оборотне перетворення бітового рядка довжиною 2n. Рядок розбивається на дві частини a і b однакової довжини в n біт. Перетворення обчислюється таким чином:

)

)

Ця операція часто використовується для «розсіювання» коду (наприклад в шифрі SAFER).

В Twofish це перетворення використовується при змішуванні результатів двох g-функцій (n = 32).

Циклічний зсув на 1 біт

У кожному раунді два правих 32-бітових блоки, які xor-яться з результатами функції F, додатково циклічно зрушуються на один біт. Третій блок зрушується до операції xor, четвертий блок – після. Ці зрушення спеціально додані, щоб порушити вирівнювання по байтах, яке властиво S-box'ам та операції множення на MDS-матрицю. Проте шифр перестає бути повністю симетричним, так як при шифруванні й розшифровці зрушення слід здійснювати в протилежні сторони.

Генерація ключів

Twofish розрахований на роботу з ключами довжиною 128, 192 і 256 біт. З вихідного ключа генерується 40 32-бітних підключів, перші вісім з яких використовуються тільки в операціях вхідного і вихідного вибілювання, а решта 32 – в раундах шифрування, по два підключі на раунд. Особливістю Twofish є те, що вихідний ключ використовується також і для зміни самого алгоритму шифрування, так як використовуються у функції g S-box'и не фіксовані, а залежать від ключа.

Для формування раундових підключів вихідний ключ M розбивається з перестановкою байт на два однакові блоки M_o і M_e. Потім за допомогою блоку M_o і функції h шифрується значення 2 * i, а за допомогою блоку M_e шифрується значення 2*i+1, де i – номер поточного раунду (0 – 15). Отримані зашифровані блоки змішуються криптоперетвореням Адамара, і потім використовуються як раундові підключі.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Інтерфейс програми зображено на рисунку 5.1.

Робота із програмою здійснюється із головного вікна. Вихідне та стиснуте зображення виводяться в окремих вікнах, розташування яких можна змінювати виходячи із погляду зручності.

Розглянемо основні інструменти управління та відображення інформації в головному вікні програми. З лівого боку вказується інформація про зображення:

– Розмір вхідного BMP-файлу.

– Розмір стиснутого файлу.

– Коефіцієнти Kicked/Zero та відсоток нульових значень пікселів при стисканні.

Справа у головному вікні користувач може встановлювати наступні параметри:

– Коефіцієнт стиснення (0-найвища якість).

– Підрозділ глибини. (Max – найбільше стиснення).

– Квантування бітів Y-UV (8, 8 – найвища якість).

Основні операції виконуються за допомогою наступних програмних кнопок:

– Відкрити.

– Стиснути.

– Зберегти.

Також ці операції доступні в меню Файл у верхній частині вікна.

За допомогою меню Вид встановлюється зовнішній вигляд програми та визначається розташування вікон із вхідним файлом та стиснутим зображенням.

На рисунку 5.2 наведено вікно довідки про програму, в якому вказується інформація про розробника, керівника, тема, дата і місце виконання.

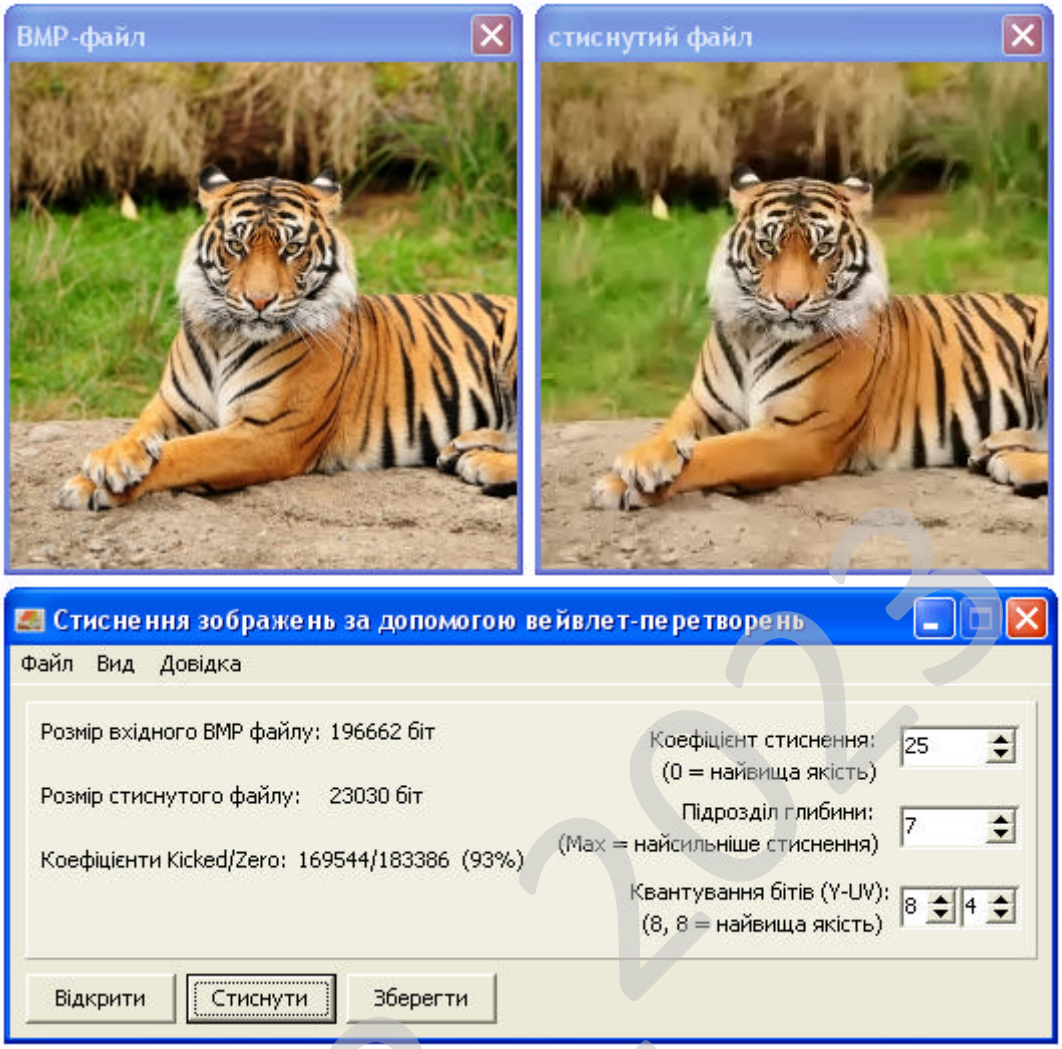

Рисунок 5.1 – Основні вікна розробленого програмного забезпечення

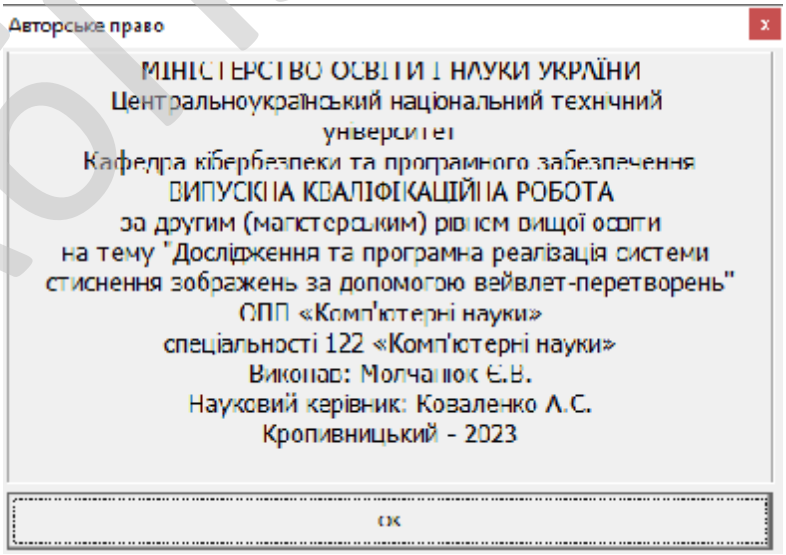

Рисунок 5.2 – Вікно довідки

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи стиснення зображень за допомогою вейвлет-перетворень.

Метою розробки є дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень.

Об'єктом дослідження є процес стиснення зображень за допомогою вейвлет-перетворень.

Предметом дослідження є методи стиснення зображень за допомогою вейвлет-перетворень.

Методи дослідження базуються на методах комп'ютерної графіки, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод стиснення зображень за допомогою вейвлет-перетворень.

– Розроблено вітчизняний продукт стиснення зображень за допомогою вейвлет-перетворень, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	260
3. Запланований термін розробки, днів	Fpq	48 (2 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	В
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	8
8. Кількість форм вихідної інформації.	–	6
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	1
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	3
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	4
17. Складність кінцевого програмного продукту (1-6)	–	5
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	3
20. Вимоги до швидкодії ПП (1-6)	–	3
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	3
24. Професійний рівень програмістів (1-6)	–	4
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	1
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	260000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	40
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+4{,}22+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}2^{1{,}027} = 5{,}5 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 5{,}5 \cdot (1 \cdot 1{,}09 \cdot 1{,}30 \cdot 0{,}91 \cdot 1 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1 \cdot 0{,}87 \cdot 1{,}10 \cdot$$

$$\cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}10 \cdot 1 \cdot 1 \cdot 1{,}10) = 12{,}9 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 12{,}9^{0{,}33+0{,}2(1{,}027-1{,}01)} \cdot 100 = 131 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	131	Ф 7.1-7.4
Впровадження	15	Д13
Всього	180	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{180 \cdot 1}{24 - 3} = 8{,}6 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	10	900	15
Монітор	60	10	600	10
Клавіатура	30	10	300	5
Маніпулятор «мишка»	30	10	300	5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	3	360	6
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	4	120	2
Кабельні господарства ЛОМ на 1 м.п.	2,5	460	1150	19,17
Копіювальний апарат	140	2	280	4,67
Усього за рік:			$З_ч$	68,17

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{68 \cdot 1}{1,2} = 57 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 57/(24 \cdot 8) = 0{,}3 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2022, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	2	0,5
	Створення графічних і стилістичних елементів сайту	1	
	Оформлення банерів і промо-сторінок	0,5	
	Розміщення графіки і контенту на Інтернет сторінках	0,5	
Всього		4	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	16329,5	32659
Продакт-менеджер	0,5	15000	15000
Інженер-програміст	8,6	16000	275200
Інженер-електронщик	0,3	15000	9000
Інженер-системотехнік	0,25	15000	7500
Адміністратор мережі	0,5	15000	15000
Системний програміст	0,25	15000	7500
Дизайнер WEB	0,5	16000	16000
Інженер-верстальник	0,25	15000	7500
Бухгалтер-економіст	0,5	16000	16000
Всього за період розробки	$R_{сп}$ = 12,65	-	$Ф_{роб}$ = 401359

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{401359}{12,65 \cdot 48} = 661 \, грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 12 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 12 \cdot 8 \cdot 20000 = 1920000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 192000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 12 \cdot 15500 = 186000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Комп'ютерторг за 15.10.23 – джерело http://computorg.ua/ru/price.html

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11186
Системний блок		6490
Процесор	AMD A8-8670E PRO (AD867BAHM44AB) AM4, 4 ядра, 4 потоки, 2.8 GHz, 3.3 GHz, TDP – 35 Вт, 28nm	
Системна плата	ASUS PRIME A320M-K сокет – AM4, DDR4, – 64 ГБ, – 3200 MHz, LAN – 1 Гбит/с, D-Sub (VGA), HDMI, 1 x M.2, 4 x Sata 6.0 Gb/s, Micro-ATX	
Відеокарта	Integrated AMD Radeon Graphics,	
Жорсткий диск	SSD 2.5" 256GB Mibrand (MI2.5SSD/CA256GBST) 256 GB, 3D TLC NAND, 2.5", SATA III (6Gb/s), швидкість читання – 560 Mb/s, швидкість запису – 470 Mb/s	
Оперативна пам’ять	DDR4 16GB 3200 MHz Dato (DT8G4DLDND32) DDR4, 8 ГБ, В наборі – 2	
DVD-привод	Не комплектується	
Корпус	Vinga CS105B-450W, Miditower, ATX, Micro – ATX, Mini – ITX, PSU – 450 Вт	
Кулер	ID-Cooling DK-01S	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1, 170/160, D-SUB, Wide)	3200
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V RUSSIA (BE525-RS)	1496

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	12	11186	13423,2	147655,2
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050

Продовження таблиці 7.7

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Сканери	1	2970	297	3267
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	169473,7

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1920000	-	-
2. Передавальні пристрої	192000	-	-
Всього по групі	2112000	5	105600
Група 4			
3. Обчислювальна техніка	169474	-	-
Всього по групі	169474	50	84737
Нематеріальні активи			
4. Нематеріальні активи	260000	10	26000

Продовження таблиці 7.8

1	2	3	4
Група 5, 6			
5. Вимірювальні пристрої	5190	25	1297,5
6. Транспортні засоби	72500	20	14500
7. Господарський інвентар	186000	25	46500
Всього по групі	263690	-	62297,5
Разом	$К_р = 2805164$		$А_р = 278634,5$

Примітка: вартість автомобіля взята по даним з автосалону автотрейдінг, вкладки автобазар, джерело http: //www.auto-trading.com.ua/sale/lot20772.html, складає 72500 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_e}, \quad (7.11)$$

де: N_e – кількість екземплярів програм, шт.

$$З_о = 661 \cdot 180/260 = 458 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 458 \cdot 10 \cdot 0{,}01 = 46 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(458+46) = 111 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 458 \cdot 15 \cdot 0{,}01 = 69 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картриджів, тонеру, грн.;

$N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо одну пачку паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 120$ грн., визначаємо вартість паперу за період розробки $N_м = 2$ міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 120 \cdot 2 = 240 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 13 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 13 грн./шт.

$$З_{М2} = 260 \cdot 13 + 13 = 3393 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{МЗ} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (240 + 3393 + 1702)/260 = 21 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 458 \cdot 15 \cdot 0{,}01 = 69 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 260 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 278635 \cdot 2/(260 \cdot 12) = 179 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$С_п = 458+46+111+69+21+69+179 = 953 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($Р_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 40%.

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де: $Р_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 40 \cdot 953 = 381 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	458
2. Додаткова зарплата виконавців	$З_д$	46
3. Відрахування на соціальні потреби	$С_{оц}$	111
4. Загальногосподарські витрати	$Г_{осп}$	69
5. Витрати на матеріали	$З_М$	21
6. Освоєння нових операційних систем, мов програмування	$О_п$	69
7. Амортизація основних фондів	$А_м$	179
8. Повна собівартість програмного забезпечення	$С_п$	953
9. Плановий прибуток	$П_р$	381
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	1334
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	266,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	1600,8

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та

пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	1601
Всього капітальних витрат	–	1601

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	34892	24424
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	801
Всього витрат за рік	$І$	34892	25225

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування системи за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 200 годин на рік до 140 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\ баз} = 200 \cdot 130 \cdot 1{,}1 \cdot 1{,}22 = 34892\ грн,$$

до:

$$З_{р\ нов} = 140 \cdot 130 \cdot 1{,}1 \cdot 1{,}22 = 24424\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$З_{ел\ баз} = З_{ел\ нов}$. Витрати на електроенергію не змінюються.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукці	50	–	1601	–	800,5
Всього відрахувань	-	–	1601	–	800,5

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - С_п) \cdot N_е - \sum_{i=1}^{m} E_{р_m} \cdot К_{р_m}, \quad (7.25)$$

де: $К_р$ – балансова вартість основних фондів розробника, грн.; $Е_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (1334 - 953) \cdot 260 - (0,05 \cdot 2112000 + 0,5 \cdot 169474 +$$
$$+0,25 \cdot 191190 + 0,2 \cdot 72500 + 0,1 \cdot 260000) \cdot 2/12 = 52621 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника.

$$T_в = \frac{2805164}{(1334 - 953) \cdot 260 \cdot 12 / 2} = 4,7 \text{ року}.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н (K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$К_б$, $К_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (34892\text{-}25225)\text{-}0,5 \cdot 1601 = 8867 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \quad (7.28)$$

$$T_{сп} = \frac{1601}{34892 - 25225} = 0,16 \text{ року}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	260
2. Повна собівартість розробленої програми	Грн.	953
3. Ціна розробленої програми	Грн.	1334
4. Плановий прибуток від реалізації розробленої програми	Грн.	381
5. Рентабельність програмної продукції	%	40
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2805164
7. Загальний прибуток від реалізації програмної продукції	Грн.	99060
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	52621
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	4.7
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1601
11.Величина економічного ефекту у користувача програмної продукції	Грн.	8867
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,16

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Загальна комп'ютеризація суспільства призвела до того, що використання комп'ютерів стало повсюдним у всіх сферах економіки та народного господарства. Застосування персональних комп'ютерів і ЕОМ дозволило значно підвищити продуктивність праці, змінити характер і зміст праці. Комп'ютеризація, поряд з незаперечними перевагами, тягне за собою і багато проблем. Для того, щоб активне застосування комп'ютерних технологій не стало додатковим чинником погіршення здоров'я, вкрай необхідно щоб робоче місце відповідало гігієнічним вимогам. Темою дипломного проекту є розробка та дослідження та реалізація програмного продукту, тому актуально буде розгляд умов праці програміста.

В охорону праці включають санітарно-гігієнічні, лікувально-профілактичні та організаційно-технічні системи правових і соціально-економічних заходів.

В кожній ІТ компанії є трудові відносини з працівниками. Згідно закону України "Про охорону праці" [1] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні;

– Положення про охорону праці;

– Накази з охорони праці;

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті

порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [1] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп’ютера на організм людини визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером

До фізичних факторів при роботі на комп’ютері можемо віднести:

– підвищену й знижену температуру повітря;

– підвищену й знижену вологість повітря;

– недостатню освітленість робочого місця;

– перевищуючі припустимі норми шуму;

– підвищений рівень іонізуючого випромінювання;

– підвищений рівень електромагнітних полів;

– підвищений рівень статичної електрики;

– небезпеку враження електричним струмом;

– бляклість екрана дисплея;

до хімічних можемо віднести:.

– виникнення, у результаті іонізації повітря при роботі комп'ютера, активних часток.

До психофізіологічні можемо віднести:.

– розумова напруга;

– втрата реальності;

– виникнення залежності;

– нервово-емоційні перевантаження;

– перенапруга зорового аналізатора.

До біологічних можемо віднести:

– бактеріологічну небезпеку пов’язану з наявністю місць з сприятливим середовищем їх розмноженням (наприклад клавіатура).

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	2
Довжина	3,5
Висота	3

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	7
Обсяг, V	$м^3$	не менше 20.0	21

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [2].

У зазначеному приміщенні працє 1 людина. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і

норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови Головного державного санітарного лікаря України [5], робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23,5	45-60	0,1
Тепла	23-25	50-70	0,1	23-25	52-70	0,11

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

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер HP Laser 107a, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5 – 28 – 2006 р можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

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

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

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

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору).

8.5 Розрахункова частина

Завдання: розрахувати *штучне освітлення робочого приміщення.*

Початкові дані: ширина *робочого* приміщення: 2 м.; довжина – 3,5 м.; висота – 3 м.

Розрахунок штучного освітлення проведемо за методом коефіцієнта використання світлового потоку.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де: *F* – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; *E = 300 Лк;*

S – площа освітлюваного приміщення (у нашому випадку *S*=2 х 3,5 = 7 $м^2$);

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

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку *K = 1,5);*

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін **($\rho_{стін}$)** і стелі **($\rho_{стелі}$),** значення коефіцієнтів дорівнюють **$\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%** *[6]***.**

Обчислимо індекс приміщення за формулою:

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

де: *S* – площа приміщення, *S* = 7 $м^2$*;*

h – розрахункова висота підвісу, *h* = 3 *м;*

A – ширина приміщення, *A = 2 м;*

B – довжина приміщення, *B =* 3,5 *м.*

Підставимо всі значення у формулу та визначемо індекса приміщення: *i=0,57.*

Знаючи індекс приміщення, за знаходимо *n = 0.29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників відповідного*

типу [6]). Підставимо всі значення у формулу, визначемо світловий потік: *F=7533 Лм.*

Для штучного освітлення приміщення використовуються *LED панель MAXUS ASSISTANCE PRO 40W 5000K WHITE (M1051440531)*, світловий потік яких $F_л = 4000$ Лм.

Число світильників визначається по формулі:

$$N=F/F_л$$

де: F – світловий потік,

$F_л$ – світловий потік одного *світильника*.

$$N= 7533/ 4000=1,9 \text{ шт.}$$

Приймаємо необхідну кількість світильників 2 шт.

Висновки до розділу

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

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного освітлення. Розроблено заходи з охорони праці.

Список використаних джерел інформації

1. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2.007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

3. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

4. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи

з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

5. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99**.** – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

6. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г.П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький: ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення: 16.06.2023).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи стиснення зображень за допомогою вейвлет-перетворень.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів стиснення зображень за допомогою вейвлет-перетворень.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем стиснення зображень за допомогою вейвлет-перетворень.

– Досліджена система стиснення зображень за допомогою вейвлет-перетворень.

– На основі отриманих результатів досліджень створена програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання стиснення зображень за допомогою вейвлет-перетворень.

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

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

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Twofish.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 8867 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,16 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Молчанюк Є.В. Дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Henry Lloyd. Interactive Computer Graphics. States Academic Press. 2022. 247 p.

3. Ranjan Parekh. Fundamentals of Image, Audio, and Video Processing Using MATLAB®With Applications to Pattern Recognition. CRC Press. 2021. 406 p.

4. Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.

5. PeterShirley, SteveMarschner. Fundamentals of Computer Graphics. 2009

6. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

7. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

8. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.

9. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. – Д.: НГУ, 2016. – 187 с.

10. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph.. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

11. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

12. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of

Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

13. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

14. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

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

16. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

17. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) – 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

18. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

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

21. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

22. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

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

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

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

26. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

27. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

28. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

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

30. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

31. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

32. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

33. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

34. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising

Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

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

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

37. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

38. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

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

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

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

42. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

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

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

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

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

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

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

49. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

50. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник наукових праць "Системи обробки інформації". – Випуск 2 (118). т.2. – Х.: ХУПС – 2014. – С. 64-67

51. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник тез VI міжнародної науково-практичної конференції "Проблеми та перспективи розвитку IT-індустрії". м. Харків. 17-18 квітня 2014р. – Харків: ХНЄУ. – 2014. – С. 240.

52. Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник. – Кіровоград: КНТУ 2013. – 257с.

53. Смірнов О.А., Дрєєв О.М., Доренський О.П. «Дослідження впливу ступеня стиснення зображень на оперативність їх доставки у телекомунікаційній системі. Збірник наукових праць "Системи обробки інформації". – Випуск 8(115). – Х.: ХУПС – 2013. – С. 234-239.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи стиснення зображень за допомогою вейвлет-перетворень.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 32-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи стиснення зображень за допомогою вейвлет-перетворень;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Visual C#.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

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

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 103 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 22.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Коваленко А.С.

Дослідження та програмна реалізація системи стиснення зображень за допомогою вейвлет-перетворень

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 25

Літера: РП

Кропивницький – 2023 року

// Program.cs - Основна програма

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Drawing;
using System.Runtime.InteropServices;
using System.Drawing.Imaging;


namespace WaveleteCompression
{
    class Program
    {
        static void Main(string[] args)
        {

            // Файл із зображенням для стиснення (реалізація алгоритму дозволяє
стискати
            // тільки квадратні зображення, розмір сторін у яких дорівнює
ступеню числа 2)
            string path =
"F:\\Projects\\WaveleteCompression\\WaveleteCompression\\bin\\Release\\test.bmp"
;

            // Копресор
            // Засікаємо час
            DateTime startTime = DateTime.Now;
            wvCompress c = new wvCompress();
            byte[] compressed = c.run(path);
            // Розрахунок витраченого часу
            TimeSpan duration = DateTime.Now - startTime;
            Console.Write("Compressed: ");
            Console.WriteLine(duration.Seconds * 1000 + duration.Milliseconds +
" ms");


            // Декомпресор
            // Засікаємо час
            startTime = DateTime.Now;
            wvDecompress d = new wvDecompress();
            byte[] decompressed = d.run(compressed);
            duration = DateTime.Now - startTime;
            Console.Write("Decompressed: ");
            Console.WriteLine(duration.Seconds * 1000 + duration.Milliseconds +
" ms");


            // Розпаковане зображення
            Bitmap bitmap1 = BytesToBitmap(decompressed);
            // Збереження розпакованого зображення
            bitmap1.Save(path + ".bmp", ImageFormat.Bmp);

            // Збереження в RAW без пост-стиску (для того, щоб можна було
поекспериментувати із пост-стиском)
            FileStream f = new System.IO.FileStream(path + ".bmp.raw",
FileMode.Create, FileAccess.Write);
```

```
            f.Write(decompressed, 0, decompressed.Length);
            f.Close();

            string ret = Console.ReadLine();

        }



        public unsafe static Bitmap BytesToBitmap(byte[] data)
        {
            Size size = new System.Drawing.Size(512, 512);
            GCHandle handle = GCHandle.Alloc(data, GCHandleType.Pinned);
            Bitmap bmp = new Bitmap(size.Width, size.Height, size.Width * 3,
PixelFormat.Format24bppRgb, handle.AddrOfPinnedObject());
            handle.Free();
            return bmp;
        }

    }
}
```

// wvCompress.cs - модуль, що реалізує компресію

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
using System.IO.Compression;
using System.IO;

namespace WaveleteCompression
{

    class wvCompress
    {

        // Константи
        public const int WV_LEFT_TO_RIGHT = 0;
        public const int WV_TOP_TO_BOTTOM = 1;

        public byte[] run(string path)
        {

            // Завантажуємо зображення з файлу
            Bitmap bmp = new Bitmap(path, true);

            // Конвертуємо завантажене зображення в байтовий масив
            byte[, ,] b = this.BmpToBytes_Unsafe(bmp);

            // Застосування вейвлету
            byte[] o = this.Compress(b, bmp.Width, bmp.Height);

            // Збереження в RAW без пост-стиску
            FileStream f = new System.IO.FileStream(path + ".raw",
FileMode.Create, FileAccess.Write);
            f.Write(o, 0, o.Length);
            f.Close();

            //Попереднє стиснення отриманого масиву звичайним Gzip-ом і
збереження у файл
            // Якщо для стиснення використовувати щось інше замість GZIP, то
можна одержати файл розміром ще в 2 рази менше
            string outGZ = path + ".gz";
            FileStream outfile = new FileStream(outGZ, FileMode.Create);
            GZipStream compressedzipStream = new GZipStream(outfile,
CompressionMode.Compress, true);
            compressedzipStream.Write(o, 0, o.Length);
            compressedzipStream.Close();

            // повертаємо нестиснений GZip-ом масив
            return o;

        }

        private byte[] Compress(byte[, ,] rgb, int c, int c)
```

```
        {
            // Значення, для квантування коефіцієнтів вейвлету
            int[] dwDiv = { 48, 32, 16, 16, 24, 24, 1, 1 };
            int[] dwTop = { 24, 32, 24, 24, 24, 24, 32, 32 };
            int SamplerDiv = 2, SamplerTop = 2;
            // Відсотки квантування Y, cr, cb компонентів кольору
            int YPerec = 100, crPerec = 85, cbPerec = 85;
            int WVCount = 6; // кількість рівнів згортки вейвлету
            // Перекодування RGB в YCrCb
            double[, ,] YCrCb = YCrCbEncode(rgb, c, c, YPerec, crPerec, cbPerec,
c, c);
            // Застосовуємо вейвлет згортку по черзі до кожного каналу кольорів
            for (int z = 0; z < 3; z++)
            {
                // Кожний канал згортаємо вказану кількість разів
                for (int dWave = 0; dWave < WVCount; dWave++)
                {
                    int wave = Convert.ToInt32(c / Math.Pow(2, dWave));
                    int wave = Convert.ToInt32(c / Math.Pow(2, dWave));
                    if (z == 2)
                    {
                        // Канал з компонентом Y квантуємо на менше значення,
                        // так як у ньому лежить структура зображення (складова
яскравості), а в інших каналах дані про кольори
                        YCrCb = WaveletePack(YCrCb, z, wave, wave, dwDiv[dWave],
dwTop[dWave], dWave);
                    }
                    else
                    {
                        YCrCb = WaveletePack(YCrCb, z, wave, wave, dwDiv[dWave]
* SamplerDiv, dwTop[dWave] * SamplerTop, dWave);
                    }
                }
            }
            // конвертація масиву в одномірний
            byte[] flattened = doPack(YCrCb, c, c, WVCount);
            return flattened;

        }

        /* Процедура впаковує масив типу Double у масив типу Byte
        За рахунок наявності в масиві великої кількості значень, що поміщаються
в межі байту.
        На початку всі Double приводяться до типу Short.
        Потім значення, що не вміщаються в тип байт дописуються в кінець
вихідного потоку, а замість них у масив байтів
        записується значення 255 */
        private byte[] doPack(double[, ,] ImgData, int c, int c, int wDepth)
        {
            short Value;
            int lPos = 0;
            int size = c * c * 3;
            // резервування для short значень
            int intCount = 0;
            short[] shorts = new short[size];
            byte[] Ret = new byte[size];
            // прохід масиву поступово по вейвлет-рівнях
            for(int d = wDepth-1; d >= 0; d-d--)
            {
                int wSize = (int)Math.Pow(2f, Convert.ToDouble(d));
```

```
int W = c / wSize;
int H = c / wSize;
int w2 = W / 2;
int h2 = H / 2;
// лівий верхній кут
if (d == wDepth - 1)
{
    for (int z = 0; z < 3; z++)
    {
        for (int j = 0; j < h2; j++)
        {
            for (int i = 0; i < w2; i++)
            {
                Value = (short)Math.Round(ImgData[z, i, j]);
                if ((Value >= -127) && (Value <= 127))
                {
                    Ret[lPos++] = Convert.ToByte(Value + 127);
                }
                else
                {
                    Ret[lPos++] = 255;
                    shorts[intCount++] = Value;
                }
            }
        }
    }
}
// правий верхній + правий нижній
for (int z = 0; z < 3; z++)
{
    for (int j = 0; j < H; j++)
    {
        for (int i = w2; i < W; i++)
        {
            Value = (short)Math.Round(ImgData[z, i, j]);
            if ((Value >= -127) && (Value <= 127))
            {
                Ret[lPos++] = Convert.ToByte(Value + 127);
            }
            else
            {
                Ret[lPos++] = 255;
                shorts[intCount++] = Value;
            }
        }
    }
}
// лівий нижній
for (int z = 0; z < 3; z++)
{
    for (int j = h2; j < H; j++)
    {
        for (int i = 0; i < w2; i++)
        {
            Value = (short)Math.Round(ImgData[z, i, j]);
            if ((Value >= -127) && (Value <= 127))
            {
                Ret[lPos++] = Convert.ToByte(Value + 127);
            }
            else
```

```
                            {
                                Ret[lPos++] = 255;
                                shorts[intCount++] = Value;
                            }
                        }
                    }
                }
            }
            // склеювання двох масивів (byte[] і short[]) в один
            int shortArraySize = intCount * 2;
            Array.Resize(ref Ret, Ret.Length + shortArraySize);
            Buffer.BlockCopy(shorts, 0, Ret, Ret.Length - shortArraySize,
shortArraySize);
            // повертаємо результуючий плоский масив
            return Ret;
        }

        private double[, ,] WaveletePack(double[, ,] ImgArray, int Component,
int c, int c, int dwDevider, int dwTop, int dwStep)
        {
            short Value;
            int cw2 = c / 2;
            int c2 = c / 2;
            // підрахунок коефіцієнта квантування
            double dbDiv = 1f / dwDevider;
            ImgArray = Wv(ImgArray, c, c, Component, WV_TOP_TO_BOTTOM);
            ImgArray = Wv(ImgArray, c, c, Component, WV_LEFT_TO_RIGHT);
            // квантування
            for (int j = 0; j < c; j++)
            {
                for (int i = 0; i < c; i++)
                {
                    if ((i >= cw2) || (j >= c2))
                    {
                        Value = (short)Math.Round(ImgArray[Component, i, j]);
                        if (Value != 0)
                        {
                            int value2 = Value;
                            if (value2 < 0) { value2 = -value2; }
                            if (value2 < dwTop)
                            {
                                ImgArray[Component, i, j] = 0;
                            }
                            else
                            {
                                ImgArray[Component, i, j] = Value * dbDiv;
                            }
                        }
                    }
                }
            }
            return ImgArray;
        }

        // Швидкий ліфтинг дискретного біортогонального CDF 9/7 вейвлету
        private double[, ,] Wv(double[, ,] ImgArray, int n, int dwCh, int
Component, int Side)
        {

            double a;
```

```
int i, j, n2 = n / 2;
double[] xWavelet = new double[n];
double[] tempbank = new double[n];

for (int dwPos = 0; dwPos < dwCh; dwPos++)
{
    if (Side == WV_LEFT_TO_RIGHT)
    {
        for (j = 0; j < n; j++) {
            xWavelet[j] = ImgArray[Component, dwPos, j];
        }
    }
    else if (Side == WV_TOP_TO_BOTTOM)
    {
        for (i = 0; i < n; i++) {
            xWavelet[i] = ImgArray[Component, i, dwPos];
        }
    }

    // Прогноз 1
    a = -1.586134342f;
    for (i = 1; i < n - 1; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }

    xWavelet[n - 1] += 2 * a * xWavelet[n - 2];

    // Оновлення 1
    a = -0.05298011854f;
    for (i = 2; i < n; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }
    xWavelet[0] += 2 * a * xWavelet[1];

    // Прогноз 2
    a = 0.8829110762f;
    for (i = 1; i < n - 1; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }
    xWavelet[n - 1] += 2 * a * xWavelet[n - 2];

    // Оновлення 2
    a = 0.4435068522f;
    for (i = 2; i < n; i += 2) {
        xWavelet[i] += a * (xWavelet[i - 1] + xWavelet[i + 1]);
    }
    xWavelet[0] += 2 * a * xWavelet[1];

    // масштаб
    a = 1f / 1.149604398f;
    j = 0;

    // множимо непарні на коефіцієнт "a"
    // ділимо парні на коефіцієнт "a"
    if (Side == WV_LEFT_TO_RIGHT)
    {
        for (i = 0; i < n2; i++) {
            ImgArray[Component, dwPos, i] = xWavelet[j++] / a;
            ImgArray[Component, dwPos, n2 + i] = xWavelet[j++] * a;
        }
```

```
                }
                else if (Side == WV_TOP_TO_BOTTOM)
                {
                    for (i = 0; i < n2; i++) {
                        ImgArray[Component, i, dwPos] = xWavelet[j++] / a;
                        ImgArray[Component, n2 + i, dwPos] = xWavelet[j++] * a;
                    }
                }

            }
            return ImgArray;
        }

        // Метод перекодування RGB в YCrCb
        private double[, ,] YCrCbEncode(byte[, ,] BytesRGB, int c, int c, double
Ydiv, double Udiv, double Vdiv, int o, int o)
        {
            double vr, vg, vb;
            double kr = 0.299, kg = 0.587, kb = 0.114, kr1 = -0.1687, kg1 =
0.3313, kb1 = 0.5, kr2 = 0.5, kg2 = 0.4187, kb2 = 0.0813;
            Ydiv = Ydiv / 100f;
            Udiv = Udiv / 100f;
            Vdiv = Vdiv / 100f;
            double[, ,] YCrCb = new double[3, c, c];
            for (int j = 0; j < o; j++)
            {
                for (int i = 0; i < o; i++)
                {
                    vb = (double)BytesRGB[0, i, j];
                    vg = (double)BytesRGB[1, i, j];
                    vr = (double)BytesRGB[2, i, j];
                    YCrCb[2, i, j] = (kr * vr + kg * vg + kb * vb) * Ydiv;
                    YCrCb[1, i, j] = (kr1 * vr - kg1 * vg + kb1 * vb + 128) *
Udiv;
                    YCrCb[0, i, j] = (kr2 * vr - kg2 * vg - kb2 * vb + 128) *
Udiv;
                }
            }
            return YCrCb;
        }



        private unsafe byte[, ,] BmpToBytes_Unsafe(Bitmap bmp)
        {
            BitmapData bData = bmp.LockBits(new Rectangle(new Point(),
bmp.Size), ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);
            // кількість байтів в bitmap
            int byteCount = bData.Stride * bmp.Height;
            byte[] bmpBytes = new byte[byteCount];
            Marshal.Copy(bData.Scan0, bmpBytes, 0, byteCount); // Скопіювання
заблокованих байтів з пам'яті

             // Не забудьте відкрити bitmap!!
            bmp.UnlockBits(bData);
            byte[, ,] ret = new byte[3, bmp.Width, bmp.Height];
            for (int z = 0; z < 3; z++)
            {
                for (int i = 0; i < bmp.Width; i++)
```

```
                {
                    for (int j = 0; j < bmp.Height; j++)
                    {
                        ret[z, i, j] = bmpBytes[j * bmp.Width * 3 + i * 3 + z];
                    }
                }
            }
            return ret;
        }

    }
}
```

```
// wvDecompress.cs - модуль, що реалізує декомпресію

using System;
using System.Collections.Generic;
using System.Text;

namespace WaveleteCompression
{
    class wvDecompress
    {

        // Константи
        public const int WV_LEFT_TO_RIGHT = 0;
        public const int WV_TOP_TO_BOTTOM = 1;

        public byte[] run(byte[] compressed)
        {
            int z;
            int dwDepth = 6; // кількість рівнів згортки вейвлету (чим їх
більше, тим краще стискається)
            // жорстко забиті розміри зображення
            int w = 512;
            int h = 512;
            // по суті розмір зображення й от цих коефіцієнтів повинні братися
із заголовку(header) стисненого файлу
            int[] dwDiv = { 48, 32, 16, 16, 24, 24, 1, 1 }, dwTop = { 24, 32,
24, 24, 24, 24, 32, 32 };
            int SamplerDiv = 2, YPerec = 100, crPerec = 85, cbPerec = 85;

            double[,,] yuv = doUnPack(compressed, w, h, dwDepth);

            // Розгорнення вейвлету
            for(z = 0; z < 2; z++)
            {
                for(int dWave = dwDepth - 1; dWave >= 0; dWave ---i)
                {
                    int w2 = Convert.ToInt32(w / Math.Pow(2, dWave));
                    int h2 = Convert.ToInt32(h / Math.Pow(2, dWave));
                    WaveleteUnPack(yuv, z, w2, h2, dwDiv[dWave] * SamplerDiv);
                }
            }
            z = 2;
            for(int dWave = dwDepth - 1; dWave >= 0; dWave ---i)
            {
                int w2 = Convert.ToInt32(w / Math.Pow(2, dWave));
                int h2 = Convert.ToInt32(h / Math.Pow(2, dWave));
                WaveleteUnPack(yuv, z, w2, h2, dwDiv[dWave]);
            }
            // YCrCb декодування + розкладання зображення в плоский масив
            byte[]  rgb_flatened = this.YCrCbDecode(yuv, w, h, YPerec, crPerec,
cbPerec);
            return rgb_flatened;
        }

        // Дана процедура є зворотньою процедурі DoPack у класі wvCompress.
        // Вона назад переводить його в (short) double-тип з типу byte[]
        private static double[, ,] doUnPack(byte[] Bytes, int c, int c, int
dwDepth)
        {
```

```
            int lPos = 0;
            byte Value;
            int intIndex = 0;
            // розмір підсумкового зображення в байтах
            int size = c * c * 3;
            // тимчасовий масив для результуючих коефіцієнтів згорнутого
вейвлету
            double[,,] ImgData = new double[3, c, c];

            int shortsLength = Bytes.Length - size;
            short[] shorts = new short[shortsLength / 2];
            Buffer.BlockCopy(Bytes, size, shorts, 0, shortsLength);

            for (int d = dwDepth - 1; d >= 0; d-d--)
            {
                int wSize = (int)Math.Pow(2, d);
                int W = c / wSize;
                int H = c / wSize;
                int w2 = W / 2;
                int h2 = H / 2;
                // лівий верхній
                if (d == dwDepth - 1)
                {
                    for (int z = 0; z < 3; z++)
                    {
                        for (int j = 0; j < h2; j++)
                        {
                            for (int i = 0; i < w2; i++)
                            {
                                Value = Bytes[lPos++];
                                if(Value == 255)
                                {
                                    ImgData[z, i, j] = shorts[intIndex++];
                                }
                                else
                                {
                                    ImgData[z, i, j] = Value - 127;
                                }
                            }
                        }
                    }
                }
                // верхній правий + нижній правий
                for (int z = 0; z < 3; z++)
                {
                    for (int j = 0; j < H; j++)
                    {
                        for (int i = w2; i < W; i++)
                        {
                            Value = Bytes[lPos++];
                            if(Value == 255)
                            {
                                ImgData[z, i, j] = shorts[intIndex++];
                            } else {
                                ImgData[z, i, j] = Value - 127;
                            }
                        }
                    }
                }
                // лівий нижній
```

```
                for (int z = 0; z < 3; z++)
                {
                    for (int j = h2; j < H; j++)
                    {
                        for (int i = 0; i < w2; i++)
                        {
                            Value = Bytes[lPos++];
                            if (Value == 255)
                            {
                                ImgData[z, i, j] = shorts[intIndex++];
                            }
                            else
                            {
                                ImgData[z, i, j] = Value - 127;
                            }
                        }
                    }
                }
            }
            // повертаємо результат
            return ImgData;
        }

        // Функція розгорнення вейвлету
        private void WaveleteUnPack(double[,,] ImgArray, int Component, int c,
int c, int dwDevider)
        {
            int cw2 = c / 2, ch2 = c / 2;
            double dbDiv = 1f / dwDevider;
            // деквантування значень
            for(int i = 0; i < c; i++)
            {
                for(int j = 0; j < c; j++)
                {
                    if ((i >= cw2) || (j >= ch2))
                    {
                        if (ImgArray[Component, i, j] != 0)
                        {
                            ImgArray[Component, i, j] /= dbDiv;
                        }
                    }
                }
            }
            // Розгорнення вейвлету
            for(int i = 0; i < c; i++)
            {
                reWv(ref ImgArray, c, Component, i, WV_LEFT_TO_RIGHT);
            }
            for(int j = 0; j < c; j++)
            {
                reWv(ref ImgArray, c, Component, j, WV_TOP_TO_BOTTOM);
            }
        }

        // Процедура зворотного швидкого ліфтингу дискретного біортогонального
CDF 9/7 вейвлету
        private void reWv(ref double[, ,] shorts, int n, int z, int dwPos, int
Side)
        {
```

```
            double a;
            double[] xWavelet = new double[n];
            double[] tempbank = new double[n];

            if(Side == WV_LEFT_TO_RIGHT)
            {
                for(int j = 0; j < n; j++)
                {
                    xWavelet[j] = shorts[z, dwPos, j];
                }
            }
            else if (Side == WV_TOP_TO_BOTTOM)
            {
                for(int  i = 0; i < n; i++)
                {
                    xWavelet[i] = shorts[z, i, dwPos];
                }
            }

            for(int i = 0; i < n / 2; i++)
            {
                tempbank[i * 2] = xWavelet[i];
                tempbank[i * 2 + 1] = xWavelet[i + n / 2];
            }
            for(int i = 0; i < n; i++)
            {
                xWavelet[i] = tempbank[i];
            }

            // відмінити масштабування
            a = 1.149604398f;
            for(int  i = 0; i < n; i++)
            {
                if(i % 2 != 0)
                {
                    xWavelet[i] = xWavelet[i] * a;
                } else {
                    xWavelet[i] = xWavelet[i] / a;
                }
            }

            // Повернути оновлення 2
            a = -0.4435068522f;
            for (int i = 2; i < n; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[0] = xWavelet[0] + 2 * a * xWavelet[1];

            // повернути прогнозування 2
            a = -0.8829110762f;
            for (int i = 1; i < n - 1; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[n - 1] = xWavelet[n - 1] + 2 * a * xWavelet[n - 2];

            // Повернути оновлення 1
```

```
            a = 0.05298011854f;
            for (int i = 2; i < n; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[0] = xWavelet[0] + 2 * a * xWavelet[1];

            // повернути прогнозування 1
            a = 1.586134342f;
            for (int i = 1; i < n - 1; i += 2)
            {
                xWavelet[i] = xWavelet[i] + a * (xWavelet[i - 1] + xWavelet[i +
1]);
            }
            xWavelet[n - 1] = xWavelet[n - 1] + 2 * a * xWavelet[n - 2];

            if(Side == WV_LEFT_TO_RIGHT)
            {
                for (int j = 0; j < n; j++)
                {
                    shorts[z, dwPos, j] = xWavelet[j];
                }
            }
            else if(Side == WV_TOP_TO_BOTTOM)
            {
                for(int i = 0; i < n; i++)
                {
                    shorts[z, i, dwPos] = xWavelet[i];
                }
            }
        }



        // Метод перекодування YCrCb в RGB
        private byte[] YCrCbDecode(double[, ,] yuv, int w, int h, double Ydiv,
double Udiv, double Vdiv)
        {
            byte[] bytes_flat = new byte[3 * w * h];
            double vr, vg, vb;
            double v, vCb, vCr;
            Ydiv = Ydiv / 100f;
            Udiv = Udiv / 100f;
            Vdiv = Vdiv / 100f;
            for(int j = 0; j < h; j++)
            {
                for (int i = 0; i < w ; i++)
                {
                    vCr = yuv[0, i, j] / Vdiv;
                    vCb = yuv[1, i, j] / Udiv;
                    v = yuv[2, i, j] / Ydiv;
                    vr = v + 1.402f * (vCr - 128f);
                    vg = v - 0.34414f * (vCb - 128f) - 0.71414f * (vCr - 128f);
                    vb = v + 1.722f * (vCb - 128f);
                    if (vr > 255) {vr = 255;}
                    if (vg > 255) {vg = 255;}
                    if (vb > 255) {vb = 255;}
```

```
                    if (vr < 0) {vr = 0;}
                    if (vg < 0) {vg = 0;}
                    if (vb < 0) {vb = 0;}
                    bytes_flat[j * w * 3 + i * 3 + 0] = (byte)vb;
                    bytes_flat[j * w * 3 + i * 3 + 1] = (byte)vg;
                    bytes_flat[j * w * 3 + i * 3 + 2] = (byte)vr;
                }
            }
            return bytes_flat;
        }

    }
}
```

// Gzip.cs - робота з архівами

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Ionic.Zip;
using System.IO;

namespace archiverZIP
{
    public partial class Gzip : Form
    {
        public ZipFile zip;
        public FolderBrowserDialog saveDialog;
        public OpenFileDialog openFiles;
        public SaveFileDialog saveFile;
        public List<string> files = new List<string>();
        public string archive;

        public Gzip()
        {
            InitializeComponent();
        }

        private void buttonChooseFiles_Click(object sender, EventArgs e)
        {
            try
            {
                openFiles = new OpenFileDialog();
                openFiles.Title = "Виберіть файли, які необхідно заархівувати";

                if (openFiles.ShowDialog() == DialogResult.OK)
                {
                    files.AddRange(openFiles.FileNames);
                }
                else return;
            }
            catch (Exception ex) { MessageBox.Show("Помилка під час вибору
файлів для архівації, спробуйте ще раз! " + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
        }

        private void buttonChooseArchive_Click(object sender, EventArgs e)
        {
            try
            {
                openFiles = new OpenFileDialog();
                openFiles.Title = "Виберіть архів, який необхідно
розархівувати";

                if (openFiles.ShowDialog() == DialogResult.OK)
                {
                    archive = openFiles.FileName;
                }
                else return;
            }
```

```
            catch (Exception ex) { MessageBox.Show("Помилка під час вибору
архіву для розархівація, спробуйте ще раз! " + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
        }

        private void buttonSaveArchive_Click(object sender, EventArgs e)
        {
            try
            {
                string path = "";

                if(radioButton1.Checked)
                    path = Application.StartupPath + "\\Archive (" +
DateTime.Now.ToShortDateString() + ").zip";
                else if(radioButton2.Checked)
                    path = Directory.GetCurrentDirectory() + "\\Archive (" +
DateTime.Now.ToShortDateString() + ").zip";
                else if(radioButton3.Checked)
                {
                    saveFile = new SaveFileDialog();
                    saveFile.Title = "Збереження архіву";
                    saveFile.FileName = "Archive (" +
DateTime.Now.ToShortDateString() + ")";
                    saveFile.Filter = "Файл ZIP|*.zip";

                    if (saveFile.ShowDialog() == DialogResult.OK)
                    {
                        path = saveFile.FileName;
                    }
                }
                else throw new Exception("Не обране місце для збереження
архіву!");

                //Створюємо об'єкт для роботи з архівом
                using (zip = new ZipFile(path, Encoding.UTF8))
                {
                    //Установлюємо рівень стиснення
                    zip.CompressionLevel = Ionic.Zlib.CompressionLevel.Default;
                    //Задаємо системну директорію TEMP для тимчасових файлів
                    zip.TempFileFolder = System.IO.Path.GetTempPath();
                    //Додаємо файл і вказуємо де він буде розташовуватися в
архіві
                    foreach (string f in files)
                    {
                        zip.AddFile(f, "\\");
                    }
                    //Зберігаємо архів
                    zip.Save();
                    zip = null;
                }

                MessageBox.Show("Дані успішно збережені", "Інфо",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            }
            catch (Exception ex) { MessageBox.Show("Помилка при спроби
збереження архіву, спробуйте ще раз! " + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
        }

        private void buttonSaveFiles_Click(object sender, EventArgs e)
```

```
        {
            try
            {
                string path = "";

                if (radioButton6.Checked)
                    path = Application.StartupPath;
                else if (radioButton5.Checked)
                    path = Directory.GetCurrentDirectory();
                else if (radioButton4.Checked)
                {
                    saveDialog = new FolderBrowserDialog();
                    saveDialog.Description = "Виберіть папку для розархівації";

                    if (saveDialog.ShowDialog() == DialogResult.OK)
                    {
                        path = saveDialog.SelectedPath;
                    }
                }
                else throw new Exception("Не обране місце для розархівації
архіву!");

                //Створюємо об'єкт для роботи з архівом
                using (zip = new ZipFile(archive, Encoding.UTF8))
                {
                    //Задаємо системну директорію TEMP для тимчасових файлів
                    zip.TempFileFolder = System.IO.Path.GetTempPath();
                    //Додаємо файл і вказуємо де він буде розташовуватися в
архіві
                    zip.ExtractAll(path,
ExtractExistingFileAction.OverwriteSilently);
                    zip = null;
                }

                MessageBox.Show("Дані успішно витягнуті", "Інфо",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            }
            catch (Exception ex) { MessageBox.Show("Помилка при спроби
збереження архіву, спробуйте ще раз! " + ex.Message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error); }
        }
    }
}
```

// Gzip.Designer.cs - робота з архівами

```
namespace archiverZIP
{
    partial class Gzip
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>

        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>

        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>

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

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>


        private void InitializeComponent()
        {
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.buttonSaveArchive = new System.Windows.Forms.Button();
            this.radioButton3 = new System.Windows.Forms.RadioButton();
            this.radioButton2 = new System.Windows.Forms.RadioButton();
            this.radioButton1 = new System.Windows.Forms.RadioButton();
            this.buttonChooseFiles = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.buttonSaveFiles = new System.Windows.Forms.Button();
            this.radioButton4 = new System.Windows.Forms.RadioButton();
            this.radioButton5 = new System.Windows.Forms.RadioButton();
            this.radioButton6 = new System.Windows.Forms.RadioButton();
            this.buttonChooseArchive = new System.Windows.Forms.Button();
            this.label2 = new System.Windows.Forms.Label();
            this.groupBox1.SuspendLayout();
            this.groupBox2.SuspendLayout();
            this.SuspendLayout();
```

```
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.buttonSaveArchive);
            this.groupBox1.Controls.Add(this.radioButton3);
            this.groupBox1.Controls.Add(this.radioButton2);
            this.groupBox1.Controls.Add(this.radioButton1);
            this.groupBox1.Controls.Add(this.buttonChooseFiles);
            this.groupBox1.Controls.Add(this.label1);
            this.groupBox1.Location = new System.Drawing.Point(11, 16);
            this.groupBox1.Margin = new System.Windows.Forms.Padding(4);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Padding = new System.Windows.Forms.Padding(4);
            this.groupBox1.Size = new System.Drawing.Size(230, 240);
            this.groupBox1.TabIndex = 0;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Архівація";

            //
            // buttonSaveArchive
            //
            this.buttonSaveArchive.Location = new System.Drawing.Point(71, 195);
            this.buttonSaveArchive.Name = "buttonSaveArchive";
            this.buttonSaveArchive.Size = new System.Drawing.Size(89, 29);
            this.buttonSaveArchive.TabIndex = 5;
            this.buttonSaveArchive.Text = "Зберегти";
            this.buttonSaveArchive.UseVisualStyleBackColor = true;
            this.buttonSaveArchive.Click += new
System.EventHandler(this.buttonSaveArchive_Click);

            //
            // radioButton3
            //
            this.radioButton3.AutoSize = true;
            this.radioButton3.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton3.Location = new System.Drawing.Point(10, 158);
            this.radioButton3.Name = "radioButton3";
            this.radioButton3.Size = new System.Drawing.Size(194, 19);
            this.radioButton3.TabIndex = 4;
            this.radioButton3.TabStop = true;
            this.radioButton3.Text = "Зберегти в обрану папку";
            this.radioButton3.UseVisualStyleBackColor = true;

            //
            // radioButton2
            //
            this.radioButton2.AutoSize = true;
            this.radioButton2.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton2.Location = new System.Drawing.Point(10, 132);
            this.radioButton2.Name = "radioButton2";
            this.radioButton2.Size = new System.Drawing.Size(184, 19);
            this.radioButton2.TabIndex = 3;
            this.radioButton2.TabStop = true;
            this.radioButton2.Text = "Зберегти в поточну папку";
            this.radioButton2.UseVisualStyleBackColor = true;
```

```
            //
            // radioButton1
            //
            this.radioButton1.AutoSize = true;
            this.radioButton1.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton1.Location = new System.Drawing.Point(10, 106);
            this.radioButton1.Name = "radioButton1";
            this.radioButton1.Size = new System.Drawing.Size(217, 19);
            this.radioButton1.TabIndex = 2;
            this.radioButton1.TabStop = true;
            this.radioButton1.Text = "Зберегти в папку із програмою";
            this.radioButton1.UseVisualStyleBackColor = true;

            //
            // buttonChooseFiles
            //
            this.buttonChooseFiles.Location = new System.Drawing.Point(127, 59);
            this.buttonChooseFiles.Name = "buttonChooseFiles";
            this.buttonChooseFiles.Size = new System.Drawing.Size(75, 23);
            this.buttonChooseFiles.TabIndex = 1;
            this.buttonChooseFiles.Text = "Огляд";
            this.buttonChooseFiles.UseVisualStyleBackColor = true;
            this.buttonChooseFiles.Click += new
System.EventHandler(this.buttonChooseFiles_Click);

            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif",
9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.label1.Location = new System.Drawing.Point(7, 32);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(210, 15);
            this.label1.TabIndex = 0;
            this.label1.Text = "Виберіть файл(ы) для архівації:";

            //
            // groupBox2
            //
            this.groupBox2.Controls.Add(this.buttonSaveFiles);
            this.groupBox2.Controls.Add(this.radioButton4);
            this.groupBox2.Controls.Add(this.radioButton5);
            this.groupBox2.Controls.Add(this.radioButton6);
            this.groupBox2.Controls.Add(this.buttonChooseArchive);
            this.groupBox2.Controls.Add(this.label2);
            this.groupBox2.Location = new System.Drawing.Point(253, 16);
            this.groupBox2.Margin = new System.Windows.Forms.Padding(4);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Padding = new System.Windows.Forms.Padding(4);
            this.groupBox2.Size = new System.Drawing.Size(230, 240);
            this.groupBox2.TabIndex = 1;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Розархівація";
```

```
            //
            // buttonSaveFiles
            //
            this.buttonSaveFiles.Location = new System.Drawing.Point(71, 195);
            this.buttonSaveFiles.Name = "buttonSaveFiles";
            this.buttonSaveFiles.Size = new System.Drawing.Size(89, 29);
            this.buttonSaveFiles.TabIndex = 9;
            this.buttonSaveFiles.Text = "Зберегти";
            this.buttonSaveFiles.UseVisualStyleBackColor = true;
            this.buttonSaveFiles.Click += new
System.EventHandler(this.buttonSaveFiles_Click);

            //
            // radioButton4
            //
            this.radioButton4.AutoSize = true;
            this.radioButton4.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton4.Location = new System.Drawing.Point(8, 158);
            this.radioButton4.Name = "radioButton4";
            this.radioButton4.Size = new System.Drawing.Size(194, 19);
            this.radioButton4.TabIndex = 8;
            this.radioButton4.TabStop = true;
            this.radioButton4.Text = "Зберегти в обрану папку";
            this.radioButton4.UseVisualStyleBackColor = true;

            //
            // radioButton5
            //
            this.radioButton5.AutoSize = true;
            this.radioButton5.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton5.Location = new System.Drawing.Point(8, 132);
            this.radioButton5.Name = "radioButton5";
            this.radioButton5.Size = new System.Drawing.Size(184, 19);
            this.radioButton5.TabIndex = 7;
            this.radioButton5.TabStop = true;
            this.radioButton5.Text = "Зберегти в поточну папку";
            this.radioButton5.UseVisualStyleBackColor = true;

            //
            // radioButton6
            //
            this.radioButton6.AutoSize = true;
            this.radioButton6.Font = new System.Drawing.Font("Microsoft Sans
Serif", 9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.radioButton6.Location = new System.Drawing.Point(8, 106);
            this.radioButton6.Name = "radioButton6";
            this.radioButton6.Size = new System.Drawing.Size(217, 19);
            this.radioButton6.TabIndex = 6;
            this.radioButton6.TabStop = true;
            this.radioButton6.Text = "Зберегти в папку із програмою";
            this.radioButton6.UseVisualStyleBackColor = true;
```

```
            //
            // buttonChooseArchive
            //
            this.buttonChooseArchive.Location = new System.Drawing.Point(128,
59);
            this.buttonChooseArchive.Name = "buttonChooseArchive";
            this.buttonChooseArchive.Size = new System.Drawing.Size(75, 23);
            this.buttonChooseArchive.TabIndex = 2;
            this.buttonChooseArchive.Text = "Огляд";
            this.buttonChooseArchive.UseVisualStyleBackColor = true;
            this.buttonChooseArchive.Click += new
System.EventHandler(this.buttonChooseArchive_Click);

            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif",
9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.label2.Location = new System.Drawing.Point(5, 32);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(215, 15);
            this.label2.TabIndex = 1;

            this.label2.Text = "Виберіть архів для розархівації:";

            //
            // Gzip
            //
            this.AutoScaleDimensions = new System.Drawing.Size(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.White;
            this.ClientSize = new System.Drawing.Size(494, 272);
            this.Controls.Add(this.groupBox2);
            this.Controls.Add(this.groupBox1);
            this.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Margin = new System.Windows.Forms.Padding(4);
            this.MaximizeBox = false;
            this.Name = "Gzip";
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Archiver ZIP";
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.groupBox2.ResumeLayout(false);
            this.groupBox2.PerformLayout();
            this.ResumeLayout(false);

        }
```

```
        #endregion

        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Button buttonSaveArchive;

        private System.Windows.Forms.RadioButton radioButton3;
        private System.Windows.Forms.RadioButton radioButton2;
        private System.Windows.Forms.RadioButton radioButton1;

        private System.Windows.Forms.Button buttonChooseFiles;

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.Button buttonSaveFiles;

        private System.Windows.Forms.RadioButton radioButton4;
        private System.Windows.Forms.RadioButton radioButton5;
        private System.Windows.Forms.RadioButton radioButton6;

        private System.Windows.Forms.Button buttonChooseArchive;
        private System.Windows.Forms.Label label2;
    }
}
```