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 растрових зображень без втрати якості”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-2
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Мікіньов В.І.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Буравченко К.О.
«____» _____ 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

Мікіньову Валерію Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи стиснення растрових зображень без втрати якості*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 33-13 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 растрових зображень без втрати якості*

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

Мікіньов В.І. Дослідження та програмна реалізація системи стиснення растрових зображень без втрати якості. 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 растрових зображень без втрати якості.

Метою розробки є дослідження та програмна реалізація системи стиснення растрових зображень без втрати якості.

Об'єктом дослідження є процес стиснення растрових зображень без втрати якості.

Предметом дослідження є методи стиснення растрових зображень без втрати якості.

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

Результат роботи – програмна реалізація системи стиснення растрових зображень без втрати якості.

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 Delphi 10.4.

Ключові слова: комп'ютерні науки, стиснення растрових зображень

ABSTRACT

Mikinov V.I. Research and software implementation of a raster image compression system without quality loss. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the system of compression of raster images without loss of quality.

The goal of the development is the research and software implementation of a system of compression of raster images without loss of quality.

The object of research is the process of compression of raster images without loss of quality.

The subject of research is the methods of compression of bitmap images without loss of quality.

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 raster image compression system without loss of quality.

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 Delphi 10.4 environment.

Keywords: computer science, compression of bitmap images

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 6
1.1 Призначення системи..... 6
1.2 Область застосування..... 7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 10
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 10
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 17
2.3 Розгорнута постановка завдання 22
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 24
3.1 Опис функціонування системи 24
3.2 Розробка структурної схеми..... 33
3.3 Розробка функціональної схеми 45
3.4 Розробка діаграми процесів..... 51
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 52
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 52
4.2 Захист розробленого програмного забезпечення..... 59
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 61
6 НАУКОВА НОВИЗНА 63

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ 64
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 64
7.2 Розрахунок трудомісткості розробки програмної продукції 66
7.3 Визначення чисельності виконавців і планового фонду зарплати 68
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника 73
7.5 Визначення собівартості розробки та ціни програмної продукції. 77
7.6 Визначення об'єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції. 80
7.7 Визначення експлуатаційних витрат 80
7.8 Визначення економічної ефективності програмної продукції. 82
7.9 Висновок 84
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 85
8.1 Вступ 85
8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером 86
8.3 Аналіз умов праці на робочому місці фахівця 87
8.4 Розробка заходів з умов поліпшення охорони праці 90
8.5 Розрахункова частина 91
9 ОСНОВНІ ВИСНОВКИ 95
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 97

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ГА	–	генетичний алгоритм
ДПФ	–	дискретне перетворення Фур'є
УБК	–	метод усіченого блокового кодування

ВСТУП

Актуальність теми. Завдання зберігання й передачі інформації в компактному виді завжди були актуальні в інформатиці. І якщо щодо текстової інформації розроблені досить ефективні методи стиску даних, то розробки якісної компресії статичних зображень і динамічної відеоінформації тільки набирають оберти. На сьогоднішній день існують і широко застосовуються стандартні методи стиску. Однак потреба в зберіганні все більших обсягів інформації й бажання передачі її по каналах зв'язку з максимальною швидкістю обумовили інтерес до дослідження й розробки більше розвинутих методів. Розвиток теорії фракталів і відповідного математичного апарата наприкінці XX в. спричинило розробку принципово нового алгоритму стиску зображень – фрактального. Даний алгоритм стиску потенційно здатний забезпечити найкраще співвідношення ступеня стиску і якості відновленого зображення.

Фрактал – це нескінченно самоподібна геометрична фігура, кожний фрагмент якої повторюється при зменшенні масштабу. Масштабна інваріантність, спостережувана у фракталах, може бути або точною, або наближеною. У більше широкому змісті під фракталами розуміють множини точок в евклідовому просторі, що мають дробову метричну розмірність (у змісті Мінковського або Хаусдорфа), або метричну розмірність, строго більшу топологічної. Термін «фрактал» був уведений Бенуа Мандельбротом в 1975 році й одержав широку популярність із виходом в 1977 році його книги «Фрактальна геометрія природи».

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи стиснення растрових зображень без втрати якості.

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

– Огляд існуючих систем стиснення растрових зображень без втрати якості.

– Дослідження системи стиснення растрових зображень без втрати якості.

– Програмна реалізація системи стиснення растрових зображень без втрати якості.

Об’єктом дослідження є процес стиснення растрових зображень без втрати якості.

Предметом дослідження є методи стиснення растрових зображень без втрати якості.

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 растрових зображень без втрати якості.

– Розроблено вітчизняний продукт стиснення растрових зображень без втрати якості, 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 растрових зображень без втрати якості.

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 растрових зображень без втрати якості, 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

Алгоритм фрактального стиску зображення відносять до алгоритмів архівування із частковою втратою інформації. Основа методу фрактального кодування – це виявлення самоподібних ділянок у зображенні. Ідея компресії зображення заснована на застосувань систем ітеруємих функцій (Iterated Function System або IFS). Уперше можливість застосування теорії IFS до проблеми стиску зображення була досліджена Майклом Барнслі. Джеквін представив метод фрактального кодування, у якому використовуються системи доменних і рангових блоків зображення (domain and range subimage blocks), блоків квадратної форми, що покривають все зображення. Цей підхід став основою для більшості методів фрактального кодування. Відповідно до методу зображення повинне бути розбите на множина таких, які не перекриваються, рангових підзображень, також визначається множина таких, які перекриваються, доменних підзображень. Для кожного рангового блоку алгоритм кодування знаходить найбільш підходящий доменний блок і афінне перетворення, що переводить цей доменний блок у даний ранговий блок. Структура зображення відображається в систему рангових блоків, доменних блоків і перетворень. Ідея полягає в наступному: припустимо, що вихідне зображення є нерухливою точкою якогось стискаючого відображення. Тоді можна замість самого зображення запам'ятати яким-небудь образом це відображення, а для відновлення досить багаторазово застосувати це відображення до будь-якого стартового зображення. По теоремі Банаха, такі ітерації завжди приводять до нерухливої точки, тобто до вихідного зображення.

1.2 Область застосування

Запропонований Барнслі метод, коротко можна описати в такий спосіб. Зображення кодується декількома простими перетвореннями, тобто визначається коефіцієнтами цих перетворень.

Наприклад, до наочних прикладів фрактальних зображень отриманих за допомогою IFS можна віднести «трикутник Серпинського» і «папороть Барнслі» рис. 1.1.

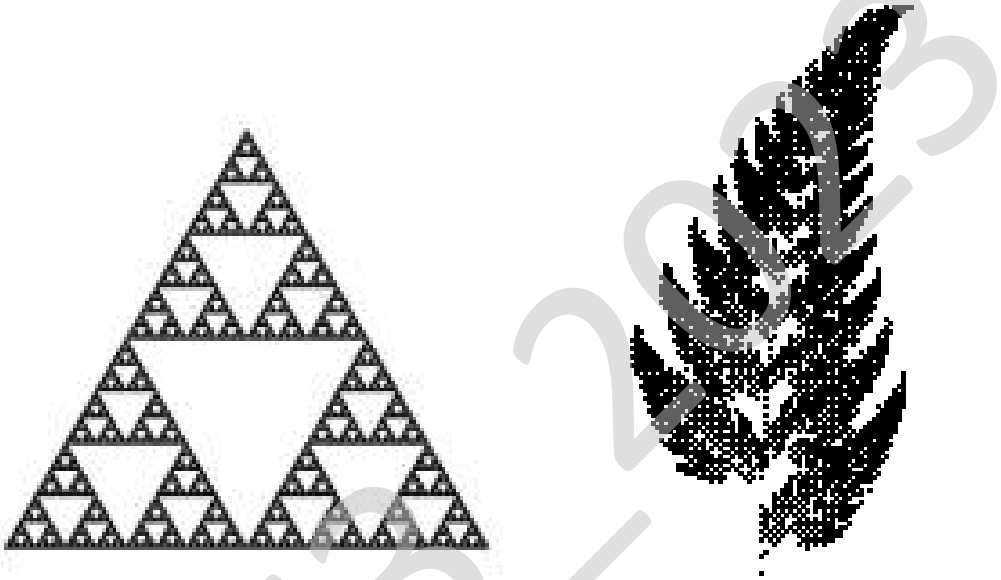

Рисунок 1.1 – «Трикутник Серпинського» і «папороть Барнслі»

Перший задається трьома, а другий чотирма афінними перетвореннями. Кожне перетворення кодується ліченими байтами, у той час як зображення, побудоване з їхньою допомогою, може займати біля мегабайта. Хоча IFS не використовуються як готові системи стиску зображень, однак вони найчастіше використовуються в різних фрактальних методах стиску.

Фрактальний алгоритм дозволяє стискати зображення в сотні й навіть тисячу разів.

Основна складність фрактального стиску полягає в тому, що для знаходження відповідних доменних блоків, потрібен повний перебір.

Оскільки при цьому переборі щораз повинні рівнятися два масиви, дана операція виходить досить тривалих, потребуючих значних обчислювальних ресурсів.

Декомпресія або відновлення вихідного зображення досить проста. Для цього необхідно виконати кілька ітерацій тривимірних афінних перетворень, коефіцієнти яких були отримані на етапі компресії. Варто відзначити, що особливість фрактального кодування полягає в тому, що декодування не залежить від дозволу.

Переважна більшість досліджень в області фрактального стиску зараз спрямовані на зменшення часу архівування, необхідного для одержання якісного зображення. На даний момент відома досить велика кількість алгоритмів оптимізації перебору, що виникає при фрактальному стиску. Для збільшення швидкості кодування велася робота із двох напрямків. Перший підхід вирішував завдання класифікації доменів (classification of domains), при якому за рахунок зменшення кількості доменів серед яких ведеться пошук, скорочується кількість обчислень. Другий підхід заснований на методі виділення особливостей (feature extraction), збільшення швидкості кодування відбувається за рахунок порівняння доменних і рангових блоків. Цікаве рішення проблеми тривалого кодування запропонував Д.С. Ватолін у своїй роботі «Використання ДКП для прискорення фрактального стиску зображень». Дискретне косинусне перетворення (ДКП) використовується для розбивки всього множини блоків у зображенні на 256 класів, що дозволяє досягти майже 100-кратного прискорення роботи алгоритму при прийнятних втратах як зображення.

На сьогоднішній день фрактальні методи щонайкраще пристосовані для додатків архівування, таких як цифрові енциклопедії, у яких кодування необхідно лише один раз, а декодування відбувається множина разів.

Фрактальні методи можна розглядати, як альтернативу технологіям, заснованим на перетворенні Фур'є, наприклад, таким як JPEG. Нові технології, такі як фрактальні повинні розглядатися не тільки як конкуренти, але і як союзники у встановленні нових стандартів.

Необхідно подальше вивчення й удосконалювання фрактальних методів, що можливо надалі повною мірою розкриє їхній потенціал.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи стиснення растрових зображень без втрати якості, 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

Графічні файли піддаються багатьом перетворенням: від елементарних, виконуваними навіть починаючими користувачами (зміни розміру, формату, ін.) і до більше складних, що виконуються фахівцями-дизайнерами, програмістами (додавання фільтрів, накладення ефектів). Прості перетворення можна зробити в стандартних програмах для перегляду зображень, тому що в багатьох з них убудовані необхідні для цього функції. Більше складну обробку роблять у конвертерах – програмах для обробки графічних зображень.

Нижче в порядку зростання описані методи пакетної обробки від простих до більше складних, професійних.

Програма Image Tuner

Найбільш проста, не ускладнена налаштуваннями й фільтрами, програма для базової обробки зображень. Перелік її можливостей обмежується зміною відтінків, розмірів, орієнтації зображення, додаванням водяного знака. Програма працює в режимі «одного вікна»: у ліву половину завантажуються файли, що піддаються обробці. У правій вказуються параметри конвертації.

Переваги цієї програми:

– простота використання;

– підтримує такі популярні формати, як JPEG, BMP, PNG, GIF, TIFF, RAW, NEF і інші;

– є функція передперегляду. Спрацьовує при кліку на зображення.

Недоліки:

– надмірна простота фільтрів. Фільтри настільки прості, що не мають налаштувань. Перебувають у розділі меню «Змінити розмір»;

– невелика кількість форматів для збереження готових ескізів: усього 5;

– недоробки в оформленні програми: частина налаштувань на англійському, що залишилася частина переведена на російську.

Таким чином, ця програма призначена для елементарної базової обробки зображень.

Програма IrfanView

Основними функціями цієї програми є перегляд зображень.

Переваги:

– доступність. Програма безкоштовна для завантаження;

– функціональність. Виконує функції перегляду й конвертації. Як конвертор працює через меню «Batch Conversion/Rename…»... Підтримує три варіанти режиму: пакетне перейменування, перетворення й змішаний режим;

– компактність (невеликий розмір);

– значна кількість форматів (близько 20);

– тестовий режим, доступний для пакетного перейменування файлів.

Недоліки:

– не всі параметри доступні для повного списку форматів;

– деякі перетворення відбуваються тільки при активації Use advanced options…», по натисканню кнопки «Advanced». В іншому випадку доступні лише стандартні для переглядача перетворення: зміна розмірів, кадрування, горизонтальне/вертикальне відбиття, водяний знак.

– передперегляд передбачається тільки для вихідного зображення. Якщо, наприклад, розміри ми можемо вказати попіксельно, то зміна колірних параметрів, яскравості, балансу являє собою проблему, тому що змінюються вони методом уведення цифр. А результати перетворень ми зможемо побачити тільки після закінчення конвертацій і виходу з «Advanced».

Таким чином, якщо вас улаштовують базові перетворення, що не вимагають передперегляду, дана програма задовольнить ваші вимоги. Але внести її в список найбільш зручних конвертерів не можна.

Програма AVS Image Converter

AVS Image Converter є програмою-конвертером. Вона є однією з набору програм AVS4You, призначених для роботи із зображеннями. Особливістю цієї програми є допоміжні утиліти Software Navigator і Update Manager. До речі, це пояснює великий розмір програми: 27 Мб.

Переваги:

– простота у використанні. Дуже простий інтерфейс, зі спрощеними функціями або передустановками, що дозволяє з легкістю працювати із програмою навіть недосвідченим користувачам;

– кількість підтримуваних форматів: 8 для запису, більше 20 для читання;

– зв'язок з Інтернетом: можна обробляти фото, імпортоване з аккаунтів Flickr або Facebook, а в самій програмі доступні передналаштування для фотознімків, призначених для публікації в Інтернеті;

– окремо розташована вкладка «Водяной знак». Застосовується для накладки тексту або зображення на фото.

Недоліки:

– мало налаштувань перетворення. В основному переважають налаштування, спрямовані на колір: яскравість, контрастність, колірна температура. Є присутнім ефект розмиття/різкості, додавання тексту. З усім переліком ефектів можна познайомитися у вкладці «Корекція».

Програма FastStone Photo Resizer

По функціональності ця програма нагадує IrfanView, але, на відміну від останньої, надає користувачеві більше зручний і розширений вибір параметрів.

Переваги:

– керування зображенням. Указавши формат вихідного зображення («Output Format») і папку збереження («Output Folder») можна заощадити час на виконанні цих функцій для кожного зображення окремо;

– функція «Search and Replace». З її допомогою назва файлів коректується без повної зміни імені.

Недоліки:

– базовий набір перетворень. Перелік доступних перетворень: коректування розміру, текстури, перспективи, додавання водяного зображення, добірка рамок;

– відсутність режиму (вікна) передперегляду;

– нераціональне застосування області інтерфейсу. Більша частина робочого простору являє собою область для вибору файлів.

Програма XnConvert

Програма XnConvert є одним з компонентів відомого багатьом користувачам переглядача зображень XnView. Створені на одній базі, програми виконують практично ідентичні дії, але все-таки мають ряд відмінностей, які будуть розглядатися нижче.

XnConvert є програмою-конвертером, призначеною тільки для перетворення зображень, і не містить у собі функцію перегляду. З однієї сторони це є плюсом програми, тому що вона виконує строго свої функції. А з іншої сторони для роботи необхідно задіяти іншу програму-переглядач.

Основні принципи роботи програми максимально прості. Зображення, що піддаються обробці, додаються методом перетаскування або за допомогою кнопок. Далі зі списку вибираються варіанти перетворення. Списки перебувають зверху й відбиваються у вигляді ескізів, що являє собою деякі незручності для користувачів. Справа в тому, що такого роду сортування не дозволяє переглянути інформацію про вихідні файли, тому що це могло бути при сортуванні у вигляді таблиці. Тому сортування є більше умовним, ніж функціональним.

Тепер більш докладно про функції перетворення. Список перетворень перебуває на основній вкладці «Дії» і ділиться на 4 групи:

– зображення: робота, спрямована на трансформацію файлу, або зв'язана із властивостями файлу;

– корекція: обробка колірної гами й рівнів;

– фільтри: размивка зображення, налаштування різкості, зміна фокуса;

– різне: інші функції.

Вкладка «Вихідні дані». У цій вкладці відбувається робота з файлами, що пройшли перетворення: вказуються параметри для збереження (ім'я, шлях збереження, формат). Є додаткові параметри для збереження, вони застосовуються для таких форматів, як GIF, PNG, JPG та ін.

Кнопка «Завантажити сценарій». Використовується для експорту зображень, призначених для роботи з Інтернетом. Упакування або відправлення по FTP або на e-mail, завантаження оброблених на Picasa– або Flickr-аккаунт зображень – це функції перебувають вище зазначеної кнопки.

І в завершення потрібно відзначити, що програма підтримує більше 500 форматів для читання. Правда, деякі з них більше специфічні, вимагають додаткових ресурсів. У цьому випадку необхідна установка GhostScript або плагіна CAD.

Програма XnView

Спрощений варіант XnConver. Простота роботи полягає в тому, що в цій програмі передбачені тільки дві вкладки. Перша – для формату й параметрів збереження. Друга містить у собі список перетворень. Для вибору й застосування перетворення необхідно додати в список половину вікна, що перебуває праворуч. Є функція «Збереження сценарію». Вікно налаштувань перебуває в меню «Інструменти – Пакетна обробка…»...

Програма «Фотоконвертер»

Програма створена в декількох редакціях: домашня (мінімальна), стандартна, професійна. Професійна редакція дозволяє використовувати функції Adobe Photoshop, підтримує більше 400 графічних форматів, функціонує з командного рядка. Стандартна версія відрізняється простотою й зручністю покрокових налаштувань, підтримує тільки самі затребувані формати зображень (JPEG, TIFF, GIF, PNG, BMP). Інтерфейс програми розділений на дві частини, тому перший крок перетворення – це додавання зображення в праву частину вікна. Другий крок – це безпосередньо саме перетворення, що відбувається через

кнопку «додати дія». У мінімальній версії, крім стандартних функцій, передбачений ефект розмивання/різкості, усунення червоних очей. Стандартна версія допускає додавання водяного знака. І третій, завершальний, крок – збереження. На цьому етапі вибирається формат збереження, додатково є опція «Перейменування файлу по масці». Завершенням і збереженням є натискання кнопки «Старт».

Хотілося б відзначити в цій програмі більше логічне розміщення функцій перетворення, ніж у розглянутій вище. Але, все таки, є певні недоробки: наприклад, функції корекції рівнів і кадрування ставляться до редагування, а в програмі вони перебувають у групі налаштувань «Автоматичні». Однак згодом при частому використанні програми це не викликає яких-небудь складностей.

Один з мінусів програми – довідка англійською мовою, але подивитися відповіді на питання, що вас цікавлять, можна на сайті.

Програма Adobe Photoshop

Дана програма є однією із затребуваних серед конвертерів. У цю програму убудовані практично всі необхідні інструменти для пакетного перетворення зображень. На даний момент актуальної є версія – CS6, поки ж розглядаємо версію CS5. Обробка файлів виробляється за допомогою екшнів, операції Batch або скрипта Image Processor.

Перший спосіб обробки зображень

Для обробки графічних зображень необхідно створити набір (Set). Для цього краще взяти тестовий зразок. Надалі цей набір буде застосований до всіх обраних файлів. Набір створюється через палітру Actions, шляхом записування необхідних дій. На даному етапі доступні тільки засоби Photoshop. Надалі список можна відкоригувати, додаючи або видаляючи певні дії. Через меню «File – Automate – Batch…»заходимо в групу налаштувань «Play», вибираємо потрібний екшн, указуємо джерело й папку для збереження оброблених файлів. Ця дія приводить до масового застосування набору.

Другий метод обробки зображень

Другий спосіб обробки є більше прийнятним, тому що не вимагає створення екшна. Метод ґрунтується на використанні скрипта Image Processor. Зміна формату або розміру зображення відбувається через меню «File – Scripts – Image Processor…»... Інші налаштування можна зробити першим способом.

Зберігаються оброблені зображення у форматах JPEG, PSD і TIFF.

Програма Adobe Lightroom

Робота програми Adobe Lightroom заснована на масовому перетворенні експортованих зображень або за допомогою модуля «Library». Для роботи із зображеннями використовуються наступні налаштування, розмічені в меню «File – Export…»:

– ExportTo-вибір експорту зображення. Як варіанти: жорсткий диск, e-mail або запис CD або DVD. Також модуль Library дозволяє експортувати зображення в Інтернет: Facebook, Flickr, Adobe Revel і SmugMug;

– Export Location – папка для збереження зображень;

– File naming – вибір ім'я файлів по масці. У програмі є великий перелік змінних. Так само змінні можуть бути взяті з метаданих зображення;

– File Settings – тут вибирається формат збереження зображень. Можна залишити вихідний формат або вибрати один з наступних: JPEG, PSD, TIFF, DNG

– Image Sizing – вказуються розміри й дозвіл збереженого зображення;

– Output Sharpening – визначення різкості зображення;

– Metadata – запит по збереженню метаданих файлу;

– Watermarking – опція накладення водяного знака.

Якщо вас не влаштовують стандартні передустановки, надані в лівій бічній панелі («Preset»), ви маєте можливість додати власні.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

− Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

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 растрових зображень без втрати якості.

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

Ідея фрактального стиску ґрунтується саме на властивості самоподоби. Але існують дві проблеми:

– по-перше, ніщо не гарантує наявність властивості самоподоби в довільного зображення;

– по-друге, якщо навіть об'єкт і є фрактальним, як виділити ту область (або області), на основі яких будується зображення.

Отже, необхідний якийсь теоретичний базис, що дозволяє вирішувати ці проблеми. Такий базис існує й наведений нижче.

Чорно-білі зображення

Метричні простори

Метрикою d у просторі X називається функція двох аргументів $d(x,y)$ така, що:

$$1) d(x,y) = d(y,x)$$
$$2) d(x,x) = 0$$
$$3) d(x,y) \le d(x,z) + d(y,z)$$
$$4) 0 < d(x,y) < +\infty$$

Тоді (X,d) – метричний простір. Послідовність точок:

$$\{x_n\} \subset X$$

називається збіжної до точки x, якщо:

$$\forall \varepsilon > 0 \exists N > 0 : \forall n \ge N \Rightarrow d(x_n, x) < \varepsilon$$

Метричний простір, де кожна послідовність Коші сходиться до точки цього простору, називається повним метричним простором**.**

Точка x називається граничною точкою множини X, якщо існує послідовність точок з X, що сходиться до точки x.

Множина у метричному просторі називається замкнутою, якщо вона містить всі свої граничні точки. Множина B з (X, d) називається обмеженою, якщо існує точка x_0 з X і кінцеве значення $R > 0$, такі, що для будь-якого x з B виконується умова $d(x,x_0)$

Компактні множини й простір Хаусдорфа

Множина C у метричному просторі (X, d) називається компактним, якщо кожна нескінченна послідовність із C має збіжну в C підпослідовність.

Визначимо простір Хаусдорфа.

Нехай (X, d) – повний метричний простір. Визначимо $H(X)$ як простір, що складається з компактних підмножин множини X. Тобто, кожна точка в $H(X)$ – це компактна підмножина з X.

Розглянемо довільну точку x з X и довільне B з $H(X)$. Покладемо по визначенню:

$$d(x, B) = \min\{d(x, y): y \in B\}.$$

У силу компактності множини B мінімум існує й обмежений. Розглянемо тепер довільні $A,B \in$ H(X) і визначимо відстань між ними як:

$$d(A, B) = \max\{d(x, B): x \in A\}.$$

Компактність А забезпечує існування й кінцівка максимуму, але $d(A, B)$ не задає метрику, тому що, у загальному випадку $d(A, B) \in d(B,A)$.

Визначимо нову міру відстані $h(A, B)$:

$$h(A, B) = \max\{d(A, B), d(B, A)\}.$$

Тепер ми одержали метрику в просторі $H(X)$. Метрика h називається метрикою Хаусдорфа, а метричний простір $(H(X), h)$ – метричним простором Хаусдорфа.

Стискаючі відображення

Перетворення $f: X \in X$ у метричному просторі (X, d) називається стискаючим відображенням, якщо існує константа s, $0 \in s < 1$ така, що:

$$d(f(x1), f(x2)) \in s*d(x_1, x_2) \text{ для всіх } x_1, x_2 \in X.$$

Константа s називається коефіцієнтом стиску відображення f. Точка x_0 називається нерухливою точкою (аттрактором) стискаючого відображення f, якщо $f(x_0) = x_0$.

Теорема про стискаючі відображення

Нехай $f : X \in X$ стискаюче відображення на повному метричному просторі (X, d). Тоді f має одну й тільки одну нерухливу точку $x_0 \in X$, і для будь-якого:

$$x_0 \in X f_n(x) \in x_0.$$

Системи ітеруємих функцій

Нехай $\{w_1, \ldots, w_N\}$ – кінцевий набір стискаючих відображень в (X, d) з коефіцієнтами стиску $s_1,\ldots, s_N$... Визначимо відображення W, на компактні множини точок з X у такий спосіб :

$$W(B) = w_1(B), \ldots, w_N(B) \text{ для кожного } B \in H(X).$$

У такий спосіб одержали стискаюче відображення $W : H(X) \in H(X)$ з коефіцієнтом стиску:

$$s = \max\{s_1, \ldots, s_N\}...$$

Система ітеруємих функцій (IFS**)** складається з повного метричного простору (X, d) і кінцевої множини стискаючих відображень $w_n : X \in X$ з коефіцієнтами стиску s_n. Коефіцієнт стиску IFS визначається як:

$$s = \max\{s_1, \ldots, s_N\}...$$

IFS будемо позначати як $\{X, w_n: n = 1,2,\ldots,N\}$...

Теорема колажу. Нехай L – точка простору $H(X)$. Задано деяке $e > 0$. Виберемо IFS $\{X, w_n: n = 1,2,\ldots,N\}$с коефіцієнтом стиску s, $0 < s < 1$, так щоб $h(L, W(L))$ *J* e, Тоді $h(L,A)$ *J* $h(L, W(L))/(1\text{-}s)$ *J* e / (1-s), де A – аттрактор даної IFS.

Афінні перетворення

Афінні перетворення – перетворення виду:

$$W\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} = \begin{pmatrix} ax + by + e \\ cx + dy + f \end{pmatrix}$$

Афінні перетворення здійснюють стиск/розтягання, перенос і поворот об'єкта.

Кодування зображень

Для одержання коефіцієнтів потрібного афінного перетворення (у найпростіших випадках) досить вирішити 2 системи їх 3-х рівнянь із 3-ма невідомими. Як приклад розглянемо побудову Трикутника Серпинського. Це зображення описується 3-ма перетвореннями :

– 1-е переводить точки {1,2,3} у точки {1,4,6};

– 2-е : {1,2,3} – в {4,2,5};

– 3-е : {1,2,3} – в {6,5,3}.

Унизу праворуч показане зображення трикутника Серпинського, отримане за допомогою імовірнісного алгоритму й перетворень знайдених зазначеним вище способом.

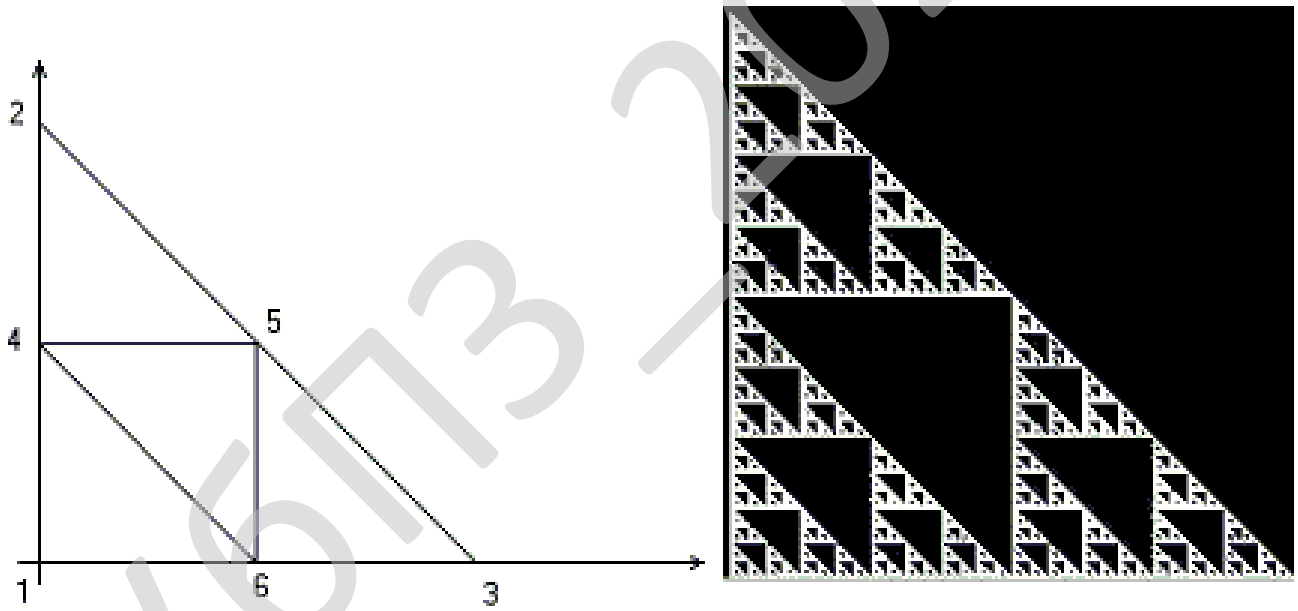

Рисунок 3.1 – Трикутник Серпинського

Декодування зображень

Детерміністичний алгоритм

Детерміністичний алгоритм для побудови зображення, що є аттрактором IFS, прямо застосовує теорему про стискаюче відображення до будь-якого

початкового зображення B з $H(X)$. Алгоритм будує послідовність зображень A_n , багаторазово застосовуючи IFS відображення $W = \{w_1, \ldots, {}_wN\}$... Якщо ми покладемо $A_0=B$, то процес може бути записаний у вигляді $A_n = W(A_{n-1})$. По теоремі про стискаюче відображення, A_n сходиться до аттрактору даного IFS. Нижче наведений приклад роботи детерміністичного алгоритму – перші кілька ітерацій і кінцеве зображення, близьке до аттрактору.

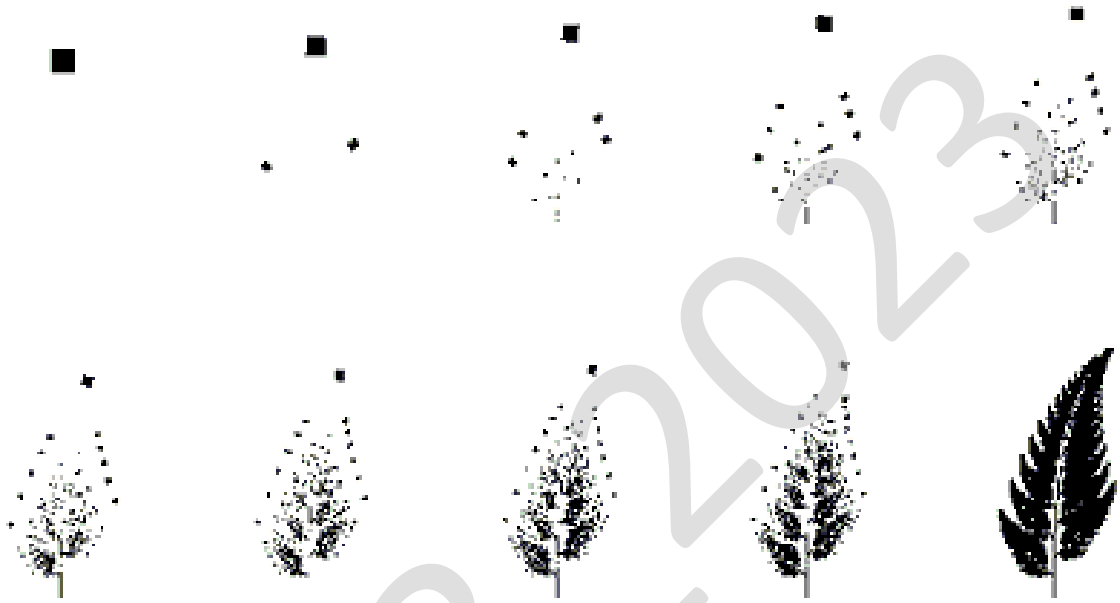

Рисунок 3.2 – Приклад роботи детерміністичного алгоритму

Імовірнісний алгоритм

У той час як детерміністичний алгоритм є прямим застосуванням теореми про стискаючі відображення, що дозволяє спостерігати, як він діє на практиці, цей алгоритм виявляється занадто повільним і звичайно не використовується на практиці для побудови зображень – аттракторів.

Більше кращим є використання імовірнісного алгоритму: Імовірнісний алгоритм зв'язує з кожним перетворенням w_i з IFS імовірність p_i. Ці ймовірності визначають, наскільки щільно кожна частина зображення – аттрактора покрита точками. Імовірності перетворень можна обчислювати як відношення модуля

визначника основної матриці перетворення до суми модулів визначників основних матриць всіх перетворень із IFS:

$$w_i(\mathbf{x}) = w_i \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} = T_i\mathbf{x} + \mathbf{t}_i,$$

$$p_i = \frac{|\det T_i|}{\sum_{j=1}^{N} |\det T_j|}.$$

Алгоритм побудови зображення:

```
For n = 1 to number_of_number_of__in_image do
   (x,y) = random(B)
   For I=1 to number_of_iterations do
      p = random(0,1)
      (x,y) = Wk(x,y) //де ймовірність p відповідає перетворенню Wk
   end
end
```

Нижче наведений приклад роботи імовірнісного алгоритму.

Рисунок 3.3 – Приклад роботи імовірнісного алгоритму

Можна помітити, що для імовірнісного алгоритму існують особливі точки – чорні зображення. Якщо робота імовірнісного алгоритму починається з них, то аттрактор не буде отриманий ніколи. Звідси випливає наступне твердження: якщо в початковому зображенні присутня хоча б одна чорна точка,

то існує відмінна від нуля ймовірність того, що в результаті роботи імовірнісного алгоритму вийде чисто чорне зображення, причому ця ймовірність тим більше, чим більше кількість чорних точок у зображенні.

Дійсно, нехай зображення має розміри *n x m*. Тоді загальне число точок у ньому $N = n \ x \ m$. Нехай число чорних точок дорівнює *b*, нехай число точок, використовуваних для побудови зображення дорівнює *M*. Імовірність того, що випадково обрана точка виявиться чорної, дорівнює *b*/*N*.

Імовірність того, що всі обрані точки виявляться чорними, дорівнює:

$$\left(\frac{b}{N}\right)^M.$$

Таким чином, для того, щоб алгоритм спрацював коректно з імовірністю більшої, ніж 1-*e*, де $e \in (0, 1)$, потрібне виконання нерівності:

$$\log_{b/N} e \in M.$$

Розглянемо приклад. Нехай $e = 0{,}01$. Тоді для коректної роботи імовірнісного алгоритму при початковому зображенні розміру 300x300 з єдиною білою точкою потрібно

$$M > \log_{89999/90000} 0.01 > 414463.$$

Цікаво, що, хоча імовірнісний алгоритм широко використовується в різних наукових працях, ніхто з авторів жодного разу не звернув уваги на даний факт.

Зображення в градаціях сірого

Метричний простір

Зображення в градаціях сірого можна розглядати як речовинні функції *f(x,y)*, визначені на одиничному квадраті:

$$I^2 = I \ x \ I.$$

На цих функціях можна ввести метрику в такий спосіб:

$$d_2(f,g) = \left(\iint_{I^2} |f(x,y) - g(x,y)|^2 \, dxdy \right)^{1/2}$$

Нехай F – простір речовинні функції, інтегрувальних із квадратом на I2 з уведеною метрикою. Тоді (F,d) – повний метричний простір і в ньому виконується теорема про стискаючі відображення.

Оскільки працювати ми будемо із цифровими зображеннями, які по суті є матрицями фіксованих значень функції $f(x,y)$, узятих у фіксованих точках (x_i, y_j), у цьому випадку можна користуватися середньоквадратичною метрикою (rms):

$$d_{rms}(f,g) = \left(\sum_{i=1}^{n} \sum_{j=1}^{m} \left| f(x_i, y_j) - g(x_i, y_j) \right|^2 \right)^{1/2}.$$

PIFS і афінні перетворення зображень у градаціях сірого

У цьому розділі використовуються IFS спеціального виду – системи ітеруємих кусочно-визначених функцій (PIFS). PIFS складається з повного метричного простору X, набору підобластей D_i з X и набору стискаючих відображень $w_i : D_i \in X$, i=1, 2, … , n...

Афінні перетворення

Нехай w_{0i} – афінне перетворення, що переводить у себе одиничний квадрат I^2. $W_{0\ i}(x,y) = A_i(x,y)T + b_i$, де A_i– деяка матриця розміру 2x2, b_i – вектор розміру 2x1. Нехай D_i – деяка підобласть I^2, і нехай R_i – область значень перетворення $w_{0\ i}$: $w_{0\ i}(D_i) = R_i$. Визначимо відображення w_i:$F \in F$, що діє на зображення $f(x,y)$:

$$w_i(f(x,y)) = s_i f(w_{0i}^{-1}(x,y)) + o_i.$$

(якщо w_{0i} – оборотно й (x,y) з R_i).

Константа s_i управляє контрастністю, o_i управляє яскравістю. w_i – це базове афінне перетворення зображень у градаціях сірого.

Стискаючі відображення

Для того, щоб відображення wі було стискаючої зажадаємо виконання умови:

$$d_2(w_i(f), w_i(g)) \ J\ s^* d_2(f, g).$$

Теорема про стискаюче відображення

Нехай $\{R_i\}$ утворять покриття множини I^2, і при цьому попарно не перетинаються. Нехай v_i – PIFS виду: $v_i : D_i \in R_i$ для деякої множини доменних областей D_i. ля кожного v_i визначимо відповідний стиск w_i на просторі зображень F зазначеним вище способом. Нехай $W(f(x,y)) = w_i(f(x,y))$ для (x, y) з R_i . Тоді W є стиском на F, має єдину нерухливу точку f і $W_n(f) \in f$ при $n \in I^2$ для будь-яких зображень f.

Теорема колажу

Нехай задане зображення в градаціях сірого f . Нехай W – стискаюче відображення таке, що $d_2(f,W(f))$ *J e*. Тоді $d_2(f, f)$ *J e*/(1-s), де s – коефіцієнт стиску відображення W, f_w – аттрактор PIFS.

Фрактальне кодування

При фрактальномукодуванні зображень ми намагаємося знайти множинустискаючих відображень, які відображають доменні блоки в множину рангових блоків.

Базовий алгоритм кодування виглядає в такий спосіб:

1. Розбиваємо вихідне зображення на непересічні рангові блоки.

2. Покриваємо зображення послідовністю доменних блоків, що можливо перетинаються (домени можуть бути різних розмірів і звичайно їхня кількість обчислюється тисячами).

3. Для кожного рангового блоку знаходимо домен і відповідне перетворення, що щонайкраще покриває ранговий блок

4. Якщо досить точної відповідності не одержуємо, то розбиваємо рангові блоки на менші. Продовжуємо цей процес доти, поки не доможемося прийнятної відповідності, або розмір рангових блоків не досягне деякого заздалегідь певної межі.

Декодування зображень

Зображення декодується шляхом ітеративного застосування перетворення W до довільного початкового зображення g, де $W(g(x,y)) = w_i(g(x,y))$, для (x,y) з R_i.

Якщо перетворення були обрані коректні , то ітерація $W_n(g)$ буде близька до вихідного значення f при деякому прийнятному значенні n. Важливо, що відповідно до теореми про стискаюче відображення процес буде сходитися незалежно від вибору початкового зображення.

3.2 Розробка структурної схеми

Фрактальний стиск зображень – це алгоритм стиску зображень с втратами, заснований на застосуванні систем ітеруємих функцій (IFS, як правило що є аффіними перетвореннями) до зображень. Даний алгоритм відомий тим, що в деяких випадках дозволяє одержати дуже високі коефіцієнти стиску (кращі приклади – до 1000 разів при прийнятній візуальній якості) для реальних фотографій природних об'єктів, що недоступно для інших алгоритмів стиску зображень у принципі. Через складну ситуацію з патентуванням широкого поширення алгоритм не одержав.

Основа методу фрактального кодування – це виявлення самоподібних ділянок у зображенні. Відповідно до даного методу зображення розбивається на безліч неперекриваючихся рангових підзображень і визначається безліч перекриваючихся доменних підзображень. Для кожного рангового блоку алгоритм кодування знаходить найбільш підходящий доменний блок і аффіне перетворення, що переводить цей доменний блок у даний ранговий блок. Структура зображення відображається в систему рангових блоків, доменних блоків і перетворень.

Ідея полягає в наступному: припустимо що вихідне зображення є нерухливою точкою якогось стискаючого відображення. Тоді можна замість самого зображення запам'ятати яким-небудь образом це відображення, а для відновлення досить багаторазово застосувати це відображення до будь-якого стартового зображення.

По теоремі Банаха, такі ітерації завжди приводять до нерухливої точки, тобто до вихідного зображення. На практиці всі труднощі полягають у

відшуканні по зображенню найбільш підходящого стискаючого відображення й у компактному його зберіганні. Як правило, алгоритми пошуку відображення (тобто алгоритми стиску) у значній мірі переборні й вимагають більших обчислювальних витрат. У той же час, алгоритми відновлення досить ефективні й швидкі.

Метод, запропонований Барнслі, можна описати в такий спосіб. Зображення кодується декількома простими перетвореннями (у нашому випадку аффінними), тобто визначається коефіцієнтами цих перетворень (у нашому випадку A, B, C, D, E, F).

Наприклад, зображення кривої Коха можна закодувати чотирма аффінними перетвореннями, ми однозначно визначимо його за допомогою всього 24-х коефіцієнтів.

Далі, поставивши чорну точку в будь-якій точці картинки ми будемо застосовувати наші перетворення у випадковому порядку якесь (досить велике) число раз (цей метод ще називають фрактальним пінг-понгом).

У результаті точка обов'язково перейде кудись усередину чорної області на вихідному зображенні. Проробивши таку операцію багато разів ми заповнимо весь чорний простір, тим самим відновивши картинку.

Основна складність фрактального стиску полягає в тому, що для знаходження відповідних доменних блоків загалом кажучи потрібен повний перебір. Оскільки при цьому переборі щораз повинні рівнятися два масиви, дана операція виходить досить тривалої. Порівняно простим перетворенням її можна звести до операції скалярного добутку двох масивів, однак навіть скалярний добуток обчислюється порівняно тривалий час.

Структурна схема системи наведена на рисунку 3.4. З нього видно, що система складається з наступних структурних блоків:

– Вхідне зображення.

– Блок параметрів стиснення.

– Блок стиснення за допомогою фракталів.

– Кодер.

– Блок запису у файл стисненого зображення.

– Блок зберігання стисненого зображення.

– Блок читання з файлу.

– Декодер.

– Блок декомпресії за допомогою фракталів.

– Вихідне зображення.

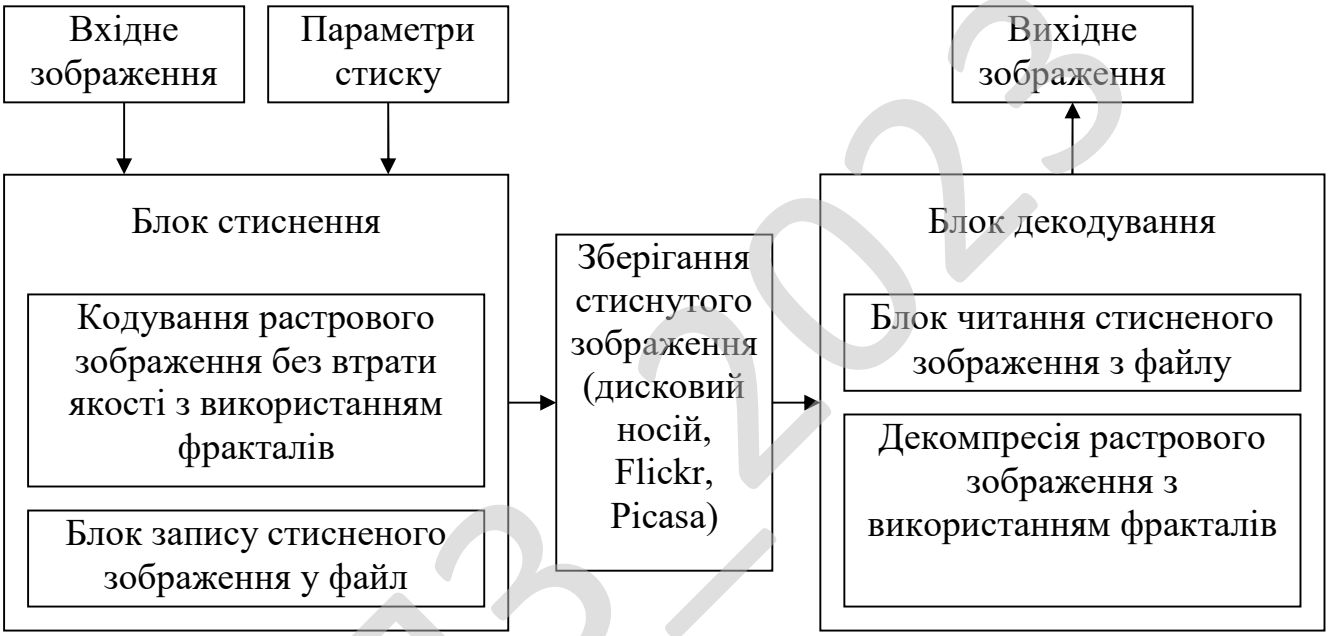

Рисунок 3.4 – Структурна схема системи

Перед тим, як перейти до докладного опису алгоритмів фрактального кодування, коротко перелічимо особливості фрактального кодування:

– У технології фрактального кодування закладений великий потенціал, але вона не стандартизована.

– Відноситься до методу стиску із втратами (дані не відновлюються у вихідному виді).

– При стиску використовуються системи ітеруємих функцій.

– Компресія даних повільна, декодування – швидке.

– Ми можемо вибирати великий діапазон значень дозволу зображень, досягаючи теоретично високих показників стиску.

– Технологія запатентована.

На сьогодні відомо кілька основних алгоритмів фрактального кодування – базовий алгоритм, алгоритм кодування FE (з виділенням характеристичних особливостей і ряд інших алгоритмів).

Зображення (у градаціях сірого) інтерпретуються як речовинні функції $f(x,y)$, певні на одиничному квадраті $I^2 = IxI$. Тобто, можна записати, що

$$f : I^2 \to \{1,2,...,N\}, \tag{3.1}$$

тут N – число відтінків сірого. Взагалі, робота із зображеннями представляється особливо трудомісткою, коли говоримо про кольорові зображення, оскільки в них точка представляється у вигляді сукупності трьох основних квітів (червоних, зелених і синього) або їхніх похідних. У зображеннях у градаціях сірого кожна точка характеризується відтінком сірого, яскравістю (діапазон від 0 до 255). Тому спочатку розглядають роботу з напівтоновими зображеннями, а потім переходять до кольорових зображень.

Можна ввести метрику на цих функціях (тому що ми працюємо з метричними просторами) у такий спосіб:

$$d_2(f,g) = \left(\int_{I^2} |f(x,y) - g(x,y)|^2 dxdy \right)^{1/2}. \tag{3.2}$$

Якщо функції безперервні, але оскільки ми будемо працювати з повним метричним простором F, певним на одиничному квадраті, а зображення є цифровими, то інтеграл убирається, співвідношення (3.9) записується в наступному вигляді:

$$d_{rms} = \left[\sum_{i=1}^{n} \sum_{j=1}^{m} |f(x_i, y_j) - g(x_i, y_j)|^2 \right]^{1/2}. \tag{3.3}$$

Цифрові зображення являють собою матрицю фіксованих значень функції $f(x,y)$, узятих у фіксованих точках $f(x_i,y_j)$. Наведене співвідношення (3.3) називається середньоквадратичним відхиленням (root meAn square). Цей показник (крім стиску зображень) може використовуватися як міра мінливості значень ознак, ступеня відхилення бажаних показників від спостережуваних. Він буде використовуватися в тому числі й для оцінки ефективності кодування зображень.

У фрактальному кодуванні використовується система ітеруємих функцій, більше загального виду. Вона називається кусочно-визначена система ітеруємих функцій (PIFS). Цей тип системи ітеруємих функцій складається з повного метричного простору X, набору підобластей і набору стискаючих відображень.

Ми також можемо визначити афінні перетворення, які переводять у себе одиничний квадрат $I^2 \rightarrow I^2$ у такий спосіб:

$$\overline{w_i}(x,y) = A_i \begin{pmatrix} x \\ y \end{pmatrix} + b_i . \qquad (3.4)$$

Тут A_i – матриці перетворень розміром $2x2$, а b_i – вектора зрушення (словом, тут звичайне афінне перетворення). А тепер можна записати відображення $w_i: F \rightarrow F$ у загальному виді.

$$\overline{w_i}(f)(x,y) = s_i f(\overline{w_i^{-1}}(x,y)) + o_i , \qquad (3.5)$$

за умови, що перетворення $\overline{w_i}$ оборотне й $(x,y) \in R_i$. Константа s_i розширює (або звужує) діапазон значень функції f, тобто управляє контрастністю (для зображень у градаціях сірого). Величина o_i відповідає за зміну яскравості зображення. Перетворення $\overline{w_i}$ називається просторовою складовою перетворення w_i. Задане співвідношення (3.5) є базовим для зображень у градаціях сірого, ми його будемо використовувати при стиску.

У випадку використання кусочно-визначеної системи ітеруємих функцій фіксована точка (або аттрактор) є зображенням f, для якої виконується $W(f)=f$. Теорема про стискаючі відображення говорить, що W у результаті буде

зображенням, а ми зможемо порахувати послідовності $W(f_0)$, $W(W(f_0))$, $W(W(W(f_0)))$, де f_0 – яке-небудь зображення. Оскільки ми знаємо, що відображення W – стискаюче на просторі зображень, то ми в результаті одержимо єдину фіксовану точку, що є якимсь зображенням.

Припустимо, що нам необхідно закодувати зображення f. Це означає, що нам необхідно визначити набір перетворень $w_1, w_2, \dots, w_N$, і при цьому:

$$W = \bigcup_{i=1}^{N} w_i, f = x_w . \tag{3.6}$$

Нам потрібно, щоб f була нерухливою точкою перетворення W. Співвідношення для фіксованих точок показує, як нам можна цього досягти:

$$f = W(f) = w_1(f) \cup w_2(f) \cup \dots \cup w_N(f). \tag{3.7}$$

Ми намагаємося знайти розбивку f на області, у результаті якого кожна область є зменшеною копією цілого зображення.

Мінімізація цього рівняння означає наступне: ми, по-перше повинні добре підібрати область D_i таким чином, щоб між областями був достатній збіг. По-друге, необхідно знайти гарні значення контрасту і яскравості (співвідношення 3.12), тобто значення коефіцієнтів s_i і o_i, для перетворення w_i. Для кожної області D_i ми зможемо порахувати, використовуючи аналітичні методи, значення коефіцієнтів, у такий спосіб досягається як можна менше значення величини d_{rms}.

Приведемо базовий алгоритм фрактального кодування.

1. Розбиваємо зображення f на непересічні рангові блоки $\{R_i\}$. Рангові блоки являють собою прямокутники (у найпростішому випадку квадрати), але можуть використовуватися й інші способи розбивки, наприклад, як рангові області можуть використовуватися трикутники. Блоки можуть бути однаковими, а може використовуватися адаптивна розбивка – методом квадродерева.

2. Покриваємо зображення послідовністю доменних блоків, що можливо перекриваються. Домени можуть бути різних розмірів, звичайно їхня кількість обчислюється сотнями й тисячами.

Домени можуть перекриватися, а можуть розміщатися на деякій відстані друг від друга. Доменне зображення темніше, ніж вихідне, оскільки яскравість при перетвореннях змінюється. Афінне перетворення записується в такий спосіб (для зображень у градаціях сірого):

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \quad (3.8)$$

Величина s_i означає яскравість пікселя й діє подібно кнопці настроювання контрасту. Коли цей параметр дорівнює 0,те пікселі домену перетворяться в чорний колір, якщо параметр дорівнює 1, то домен залишається таким же, якщо параметр установлений у межах від 0 до 1 (найчастіше беруть 0.75), чим значення більше, тим більше контраст області. Параметр o_i відповідає за світність пікселя і його зміна аналогічно зміні настроюванню яскравості. Позитивні значення роблять блок пікселів більше світлим, негативні значення – більше темним. Коли їсти можливість контролювати показники контрасту і яскравості, то можна здійснити розширене афінне перетворення, що відображає доменні блоки в рангові блоки.

Матрична частина перетворення відповідає за поворот і зміну яскравості зображення (коефіцієнти A_i, b_i, c_i, d_i і s_i), а вектор [e_i, f_i, o_i] за зрушення й зміну контрасту.

Зміна яскравості доменного зображення відбувається в такий спосіб: після його одержання яскравість всіх пікселів множиться на деяку величину, розповсюджене значення – 0.75.

3. Для кожного рангового блоку знаходимо домен і відповідне перетворення, що щонайкраще перекриває ранговий блок. Настроюються параметри перетворення – яскравість і контраст, що забезпечує найкращу відповідність.

При використанні даного підходу до кодування розмір доменних блоків завжди вдвічі більше, ніж розмір регіонів. Якщо ранговий блок має розмір 8*x*8, то домен завжди 16*x*16.

У результаті кодування у файл записується код зображення:

1. Кількість регіонів по горизонталі й вертикалі.
2. Розмір регіону.
3. Коефіцієнти афінних перетворень:

– Координати домену.

– Номер афінного перетворення.

– Різницю усередненої яскравості між регіоном і доменом.

У файл, таким чином, зберігаються тільки числові коефіцієнти, а не саме зображення. Чисел досить для розпакування (виконання зворотних перетворень, при декодуванні 2 і 4 перетворення міняються місцями).

Одним із ключових параметрів є зсув доменів (домени можуть перекриватися, а можуть розташовуватися друг від друга на деякій відстані). Процес фрактального стиску полягає в пошуку самоподібних областей зображення. У цьому випадку послідовно перебираються всі регіони, і для кожного регіону підбирається найбільш схожий на нього доменний блок. Таким чином, застосовується сукупність перетворень: порівняння блоків по пікселям і афінні перетворення. Найпоширеніші наступні:

1. Поворот на 0 градусів.
2. Поворот на 90 градусів.
3. Поворот на 180 градусів.
4. Поворот на 270 градусів.
5. Симетрія щодо осі *X*.
6. Симетрія щодо осі *Y*.
7. Симетрія щодо головної діагоналі.
8. Симетрія щодо побічної діагоналі.

Від характеру вихідного зображення залежить, які перетворення можна виконувати. Можна вибрати всі перетворення, а можна – виконати тільки перетворення повороту або тільки симетрію. Але рекомендується виконати всі 8 перетворень.

Характер використання зображень задає ступінь важливості наступних нижче суперечливих вимог до алгоритму:

1. Високий ступінь компресії. Помітимо, що далеко не для всіх додатків актуальний високий ступінь компресії. Крім того, деякі алгоритми дають краще співвідношення якості до розміру файлу при високих ступенях компресії, однак програють іншим алгоритмам при низьких ступенях.

2. Висока якість зображень. Виконання цієї вимоги прямо суперечить виконанню попередньої.

3. Висока швидкість компресії. Ця вимога для деяких алгоритмів із втратою інформації є взаємовиключаючою з першими двома. Інтуїтивно зрозуміло, що чим більше часу ми будемо аналізувати зображення, намагаючись одержати найвищий ступінь компресії, тим краще буде результат. І, відповідно, чим менше ми часу витратимо на компресію (аналіз), тим нижче буде якість зображення й більше його розмір.

4. Висока швидкість декомпресії. Досить універсальна вимога, актуальне для багатьох додатків. Однак можна привести приклади додатків, де час декомпресії далеко не критичний.

5. Масштабування зображень. Дана вимога має на увазі легкість зміни розмірів зображення до розмірів вікна активного додатка. Справа в тому, що одні алгоритми дозволяють легко масштабувати зображення прямо під час декомпресії, у той час як інші не тільки не дозволяють легко масштабувати, але й збільшують імовірність появи неприємних артефактів після застосування стандартних алгоритмів масштабування до декомпресованого зображення. Наприклад, можна привести приклад “поганого” зображення для алгоритму

JPEG – це зображення з досить дрібним регулярним рисунком (піджак у дрібну клітку). Характер внесених алгоритмом JPEG перекручувань такий, що зменшення або збільшення зображення може дати неприємні ефекти.

6. Можливість показати огрублене зображення (низького розрішення), використавши тільки початок файлу. Дана можливість актуальна для різного роду мережних додатків, де перекачування зображень може зайняти досить великий час, і бажано, одержавши початок файлу, коректно показати preview. Помітимо, що примітивна реалізація зазначеної вимоги шляхом записування в початок зображення його зменшеної копії помітно погіршить ступінь компресії.

7. Стійкість до помилок. Дана вимога означає локальність порушень у зображенні при псуванні або втраті фрагмента переданого файлу. Дана можливість використовується при широкомовленні зображень по мережі, тобто в тих випадках, коли неможливо використовувати протокол передачі, повторно запитуючі дані в сервера при помилках. Наприклад, якщо передається відеоряд, то було б неправильно використовувати алгоритм, у якого збій приводив би до припинення правильного показу всіх наступних кадрів. Дана вимога суперечить високому ступеню архівування, оскільки інтуїтивно зрозуміло, що ми повинні вводити в потік надлишкову інформацію. Однак для різних алгоритмів обсяг цієї надлишкової інформації може істотно відрізнятися.

8. Облік специфіки зображення. Більше високий ступінь архівування для класу зображень, які статистично частіше будуть застосовуватися в нашому додатку. У попередніх розділах ця вимога вже обговорювалася.

9. Редактуємість. Під редактуємістю розуміється мінімальний ступінь погіршення якості зображення при його повторному збереженні після редагування. Багато алгоритмів із втратою інформації можуть істотно зіпсувати зображення за кілька ітерацій редагування.

10. Невелика вартість апаратної реалізації. Ефективність програмної реалізації. Дані вимоги до алгоритму реально пред'являють не тільки виробники

ігрових приставок, але й виробники багатьох інформаційних систем. Так, декомпресор фрактального алгоритму дуже ефективно й коротко реалізується з використанням технології MMX і розпаралелювання обчислень, а стиск по стандарту CCITT Group 3 легко реалізується апаратно.

Очевидно, що для конкретного завдання нам будуть дуже важливі одні вимоги й менш важливі (і навіть абсолютно байдужні) інші.

На практиці для кожного завдання ми можемо сформулювати набір пріоритетів з вимог, викладених вище, що і визначить найбільш підходящий у наших умовах алгоритм (або набір алгоритмів) для її рішення.

Критерії порівняння алгоритмів

Помітимо, що характеристики алгоритму щодо деяких вимог додатків, сформульовані вище, залежать від конкретних умов, у які буде поставлений алгоритм. Так, ступінь компресії залежить від того, на якому класі зображень алгоритм тестується. Аналогічно, швидкість компресії нерідко залежить від того, на якій платформі реалізований алгоритм. Перевага одному алгоритму перед іншим може дати, наприклад, можливість використання в обчисленнях алгоритму технологій нижнього рівня, типу MMX, а це можливо далеко не для всіх алгоритмів. Так, JPEG істотно виграє від застосування технології MMX, а LZW ні. Крім того, нам доведеться враховувати, що деякі алгоритми розпаралелюються легко, а деякі ні.

Таким чином, неможливо скласти універсальний порівняльний опис відомих алгоритмів. Це можна зробити тільки для типових класів додатків за умови використання типових алгоритмів на типових платформах. Однак такі дані надзвичайно швидко застарівають.

У той же час ми можемо розглянути таку рідку на сьогодні вимогу, як стійкість до помилок. Можна припустити, що незабаром (через 5-10 років) з поширенням широкомовлення в мережі Internet для його забезпечення будуть

використовуватися саме алгоритми, стійкі до помилок, навіть не розглянуті в сьогоднішніх статтях і оглядах.

З усіма зробленими вище застереженнями, виділимо декілька найбільш важливих для нас критеріїв порівняння алгоритмів компресії, які й будемо використовувати надалі. Як легко помітити, ми будемо обговорювати менше критеріїв, ніж було сформульовано вище:

1. Гірший, середній і кращий коефіцієнти стиску. Тобто частка, на яку зросте зображення, якщо вихідні дані будуть найгіршими; якийсь середньостатистичний коефіцієнт для того класу зображень, на який орієнтований алгоритм; і, нарешті, кращий коефіцієнт. Останній необхідний лише теоретично, оскільки показує ступінь стиску найкращого (як правило, абсолютно чорного) зображення, іноді фіксованого розміру.

2. Клас зображень, на який орієнтований алгоритм. Іноді зазначено також, чому на інших класах зображень виходять гірші результати.

3. Симетричність. Відношення характеристики алгоритму кодування до аналогічної характеристики при декодуванні. Характеризує ресурсоємність процесів кодування й декодування. Для нас найбільш важливою є симетричність за часом: відношення часу кодування вчасно декодування. Іноді нам буде потрібно симетричність по пам'яті.

4. Є чи втрати якості? І якщо є, то за рахунок чого змінюється коефіцієнт архівування? Справа в тому, що в більшості алгоритмів стиску із втратою інформації існує можливість зміни коефіцієнта стиску.

5. Характерні риси алгоритму й зображень, до яких його застосовують. Тут можуть вказуватися найбільш важливі для алгоритму властивості, які можуть стати визначальними при його виборі.

3.3 Розробка функціональної схеми

У рамках комп'ютерної графіки бурхливо розвивається зовсім нова область – алгоритми архівування зображень. Поява цієї області обумовлене тим, що зображення – це своєрідний тип даних, характеризуємий трьома особливостями:

1. Зображення (як і відео) займають набагато більше місця в пам'яті, ніж текст. Так, скромна, не дуже якісна ілюстрація на обкладинці книги розміром 500x800 точок, займає 1.2 Мб – стільки ж, скільки художня книга з 400 сторінок (60 знаків у рядку, 42 рядка на сторінці). Як приклад можна розглянути також, скільки тисяч сторінок тексту ми зможемо помістити на CD-ROM, і як мало там поміститься якісних незжатих фотографій. Ця особливість зображень визначає актуальність алгоритмів архівування графіки.

2. Другою особливістю зображень є те, що людський зір при аналізі зображення оперує контурами, загальним переходом кольорів і порівняно невідчутно до малих змін у зображенні. Таким чином, ми можемо створити ефективні алгоритми архівування зображень, у яких декомпресоване зображення не буде збігатися з оригіналом, однак людина цього не помітить. Дана особливість людського зору дозволила створити спеціальні алгоритми стиску, орієнтовані тільки на зображення. Ці алгоритми мають дуже високі характеристики.

3. Ми можемо легко помітити, що зображення, у відмінність, наприклад, від тексту, має надмірність в 2-х вимірах. Тобто як правило, сусідні точки, як по горизонталі, так і по вертикалі, у зображенні близькі за кольором. Крім того, ми можемо використовувати подобу між колірними площинами R, G і B у наших алгоритмах, що дає можливість створити ще більш ефективні алгоритми. Таким чином, при створенні алгоритму компресії графіки ми використовуємо особливості структури зображення.

На рисунку 3.5 зображена функціональна схема системи. Нижче

розглянемо її більш докладно.

Схема складається з двох великих функціональних блоків:

– Блок компресії.

– Блок декомпресії.

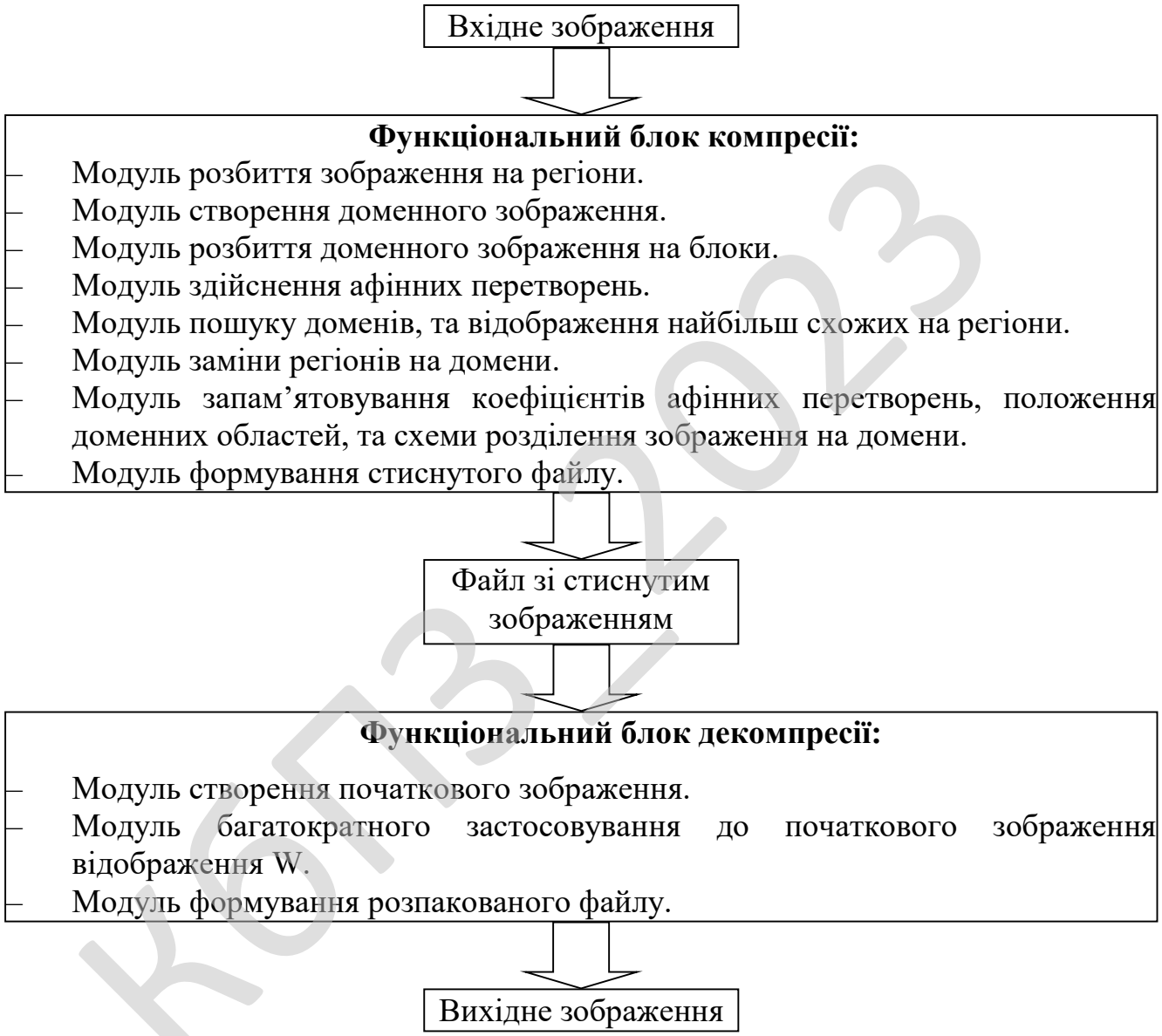

Рисунок 3.5 – Функціональна схема системи

На блок компресії подається вхідне зображення, яке необхідно стиснути.

Після цього це зображення послідовно, у блоці компресії, обробляється наступними модулями:

– Модуль розбиття зображення на регіони.

– Модуль створення доменного зображення.

– Модуль розбиття доменного зображення на блоки.

– Модуль здійснення афінних перетворень.

– Модуль пошуку доменів, та відображення найбільш схожих на регіони.

– Модуль заміни регіонів на домени.

– Модуль запам’ятовування коефіцієнтів афінних перетворень, положення доменних областей, та схеми розділення зображення на домени.

– Модуль формування стиснутого файлу.

Функціональний блок декомпресії складається з наступних модулів:

– Модуль створення початкового зображення.

– Модуль багатократного застосовування до початкового зображення відображення W.

– Модуль формування розпакованого файлу.

Після того, як стиснуте зображення буде піддано послідовній обробці вищеперерахованих модулів, отримається вихідне зображення.

Класи зображень

Статичні растрові зображення являють собою двовимірний масив чисел. Елементи цього масиву називають пікселями. Всі зображення можна підрозділити на дві групи – з палітрою й без неї. У зображень із палітрою в пікселі зберігається число – індекс у деякому одномірному векторі кольорів, називаному палітрою. Найчастіше зустрічаються палітри з 16 і 256 кольорів.

Зображення без палітри бувають у якій-небудь системі кольоропредставлення й у градаціях сірого (grayscale). Для останніх значення кожного пікселя інтерпретується як яскравість відповідної точки. Зустрічаються зображення з 2, 16 і 256 рівнями сірого. Одне із цікавих практичних завдань полягає в приведенні кольорового або чорно-білого зображення до двох градацій яскравості, наприклад, для печатки на лазерному принтері. При використанні якоїсь системи кольоропредставлення кожний піксель являє собою запис (структуру), полями якої є компоненти кольору. Найпоширенішої є система RGB,

у якій колір представлений значеннями інтенсивності червоної (R), зеленої (G) і синьої (B) компонент. Існують і інші системи кольоропредставлення, такі, як CMYK, CIE XYZccir 60-1 і т.п. Нижче ми побачимо, як використовуються колірні моделі при стиску зображень із втратами.

Для того, щоб коректніше оцінювати ступінь стиску, потрібно ввести поняття класу зображень. Під класом буде розумітися якась сукупність зображень, застосування до яких алгоритму архівування дає якісно однакові результати. Наприклад, для одного класу алгоритм дає дуже високий ступінь стиску, для іншого – майже не стискає, для третього – збільшує файл у розмірі. (Відомо, що багато алгоритмів у найгіршому разі збільшують файл.)

Розглянемо наступні приклади неформального визначення класів зображень:

1. Клас 1. Зображення з невеликою кількістю кольорів (4-16) і більшими областями, заповненими одним кольором. Плавні переходи кольорів відсутні. Приклади: ділова графіка – гістограми, діаграми, графіки й т.п.

2. Клас 2. Зображення, із плавними переходами кольорів, побудовані на комп'ютері. Приклади: графіка презентацій, ескізні моделі в САПР, зображення, побудовані по методу Гуро.

3. Клас 3. Фотореалістичні зображення. Приклад: відсканованні фотографії.

4. Клас 4. Фотореалістичні зображення з накладенням ділової графіки. Приклад: реклама.

Розвиваючи дану класифікацію, як окремі класи можуть бути запропоновані неякісно відскановані в 256 градацій сірого кольору сторінки книг або растрові зображення топографічних карт. (Помітимо, що цей клас не тотожний класу 4). Формально будучи 8– або 24-бітними, вони несуть навіть не растрову, а чисто векторну інформацію. Окремі класи можуть утворювати й зовсім специфічні зображення: рентгенівські знімки або фотографії в профіль і фас із електронного досьє. Досить складним і цікавим завданням є пошук найкращого алгоритму для конкретного класу зображень.

Нема рації говорити про те, що якийсь алгоритм стиску краще іншого, якщо ми не позначили класи зображень, на яких рівняються наші алгоритми.

Класи додатків

Розглянемо наступну просту класифікацію додатків, що використовують алгоритми компресії:

1. Клас 1. Характеризуються високими вимогами вчасно архівування й розархівування. Нерідко потрібен перегляд зменшеної копії зображення й пошук у базі даних зображень. Приклади: Видавничі системи в широкому змісті цього слова. Причому якісні публікації, що як готовлять (журнали) зі свідомо високою якістю зображень і використанням алгоритмів архівування без втрат, так і газети, що готовлять, і інформаційні вузли в WWW, де є можливість оперувати зображеннями меншої якості й використовувати алгоритми стиску із втратами. У подібних системах доводиться мати справа з повнокольоровими зображеннями самого різного розміру (від 640x480 – формат цифрового фотоапарата, до 3000x2000) і з великим двоцвітними зображеннями. Оскільки ілюстрації займають левову частину від загального обсягу матеріалу в документі, проблема зберігання коштує дуже гостро. Проблеми також створює більша різнорідність ілюстрацій (доводиться використовувати універсальні алгоритми). Єдине, що можна сказати заздалегідь, це те, що будуть переважати фотореалістичні зображення й ділова графіка.

2. Клас 2. Характеризується високими вимогами до ступеня архівування й часу розархівування. Час архівування ролі не грає. Іноді подібні додатки також жадають від алгоритму компресії легкості масштабування зображення під конкретний дозвіл монітора в користувача. Приклад: Довідники й енциклопедії на CD-ROM. При створенні енциклопедій і ігор більшу частину диска займають статичні зображення й відео. Таким чином, для цього класу додатків актуальність здобувають істотно асиметричні за часом алгоритми (симетричність за часом – відношення часу компресії вчасно декомпресії).

3. Клас 3. Характеризується дуже високими вимогами до ступеня архівування. Додаток клієнта одержує від сервера інформацію з мережі. Приклад: “Всесвітня інформаційна павутина” – WWW. У цій гіпертекстовій системі досить активно використовуються ілюстрації. При оформленні інформаційних або рекламних сторінок хочеться зробити їх більше яскравими й барвистими, що природно позначається на розмірі зображень. Найбільше при цьому страждають користувачі, підключені до мережі за допомогою повільних каналів зв'язку. Якщо сторінка WWW перенасичена графікою, то очікування її повної появи на екрані може затягтися. Оскільки при цьому навантаження на процесор мале, то тут можуть знайти застосування ефективно стискаючі складні алгоритми з порівняно більшим часом розархівування. Крім того, ми можемо видозмінити алгоритм і формат даних так, щоб переглядати огрублене зображення файлу до його повного одержання. Можна привести безліч більше вузьких класів додатків. Так, своє застосування машинна графіка знаходить і в різних інформаційних системах. Наприклад, уже стає звичним досліджувати ультразвукові й рентгенівські знімки не на папері, а на екрані монітора. Поступово в електронний вид переводять і історії хвороб. Зрозуміло, що зберігати ці матеріали логічніше в єдиній картотеці. При цьому без використання спеціальних алгоритмів більшу частину архівів займуть фотографії. Тому при створенні ефективних алгоритмів рішення цього завдання потрібно врахувати специфіку рентгенівських знімків – перевагу розмитих ділянок. Це неминуче накладає свої обмеження на алгоритм компресії. В електронних картотеках і досьє різних служб для зображень характерна подоба між фотографіями в профіль, і подоба між фотографіями у фас, що також необхідно враховувати при створенні алгоритму архівування. Подоба між фотографіями спостерігається й у будь-яких інших спеціалізованих довідниках. Як приклад можна привести енциклопедії птахів або квітів. Нема рації говорити про те, що якийсь конкретний алгоритм компресії краще іншого, якщо ми не позначили клас додатків, для якого ми ці алгоритми збираємося порівнювати.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.6. Після початку роботи розробленого ПЗ ми потрапляємо до головного вікна ПЗ звідки можемо перейти до налаштувань ПЗ, бібліотеки системи стиснення растрових зображень без втрати якості провести відкриття стиснутого файлу та через аналіз зображення спробувати декомпресувати файл, провести виведення розпакованого зображення на екран, вибрати формат зображення та зберегти файл у вказаному форматі. Також через головне вікно ПЗ можемо провести відкриття файлу зображення та за допомогою аналізу зображення провести компресію файлу з попереднім переглядом результатів перед збереженням та подальшою компресією зображення, виведенням зображення на екран.

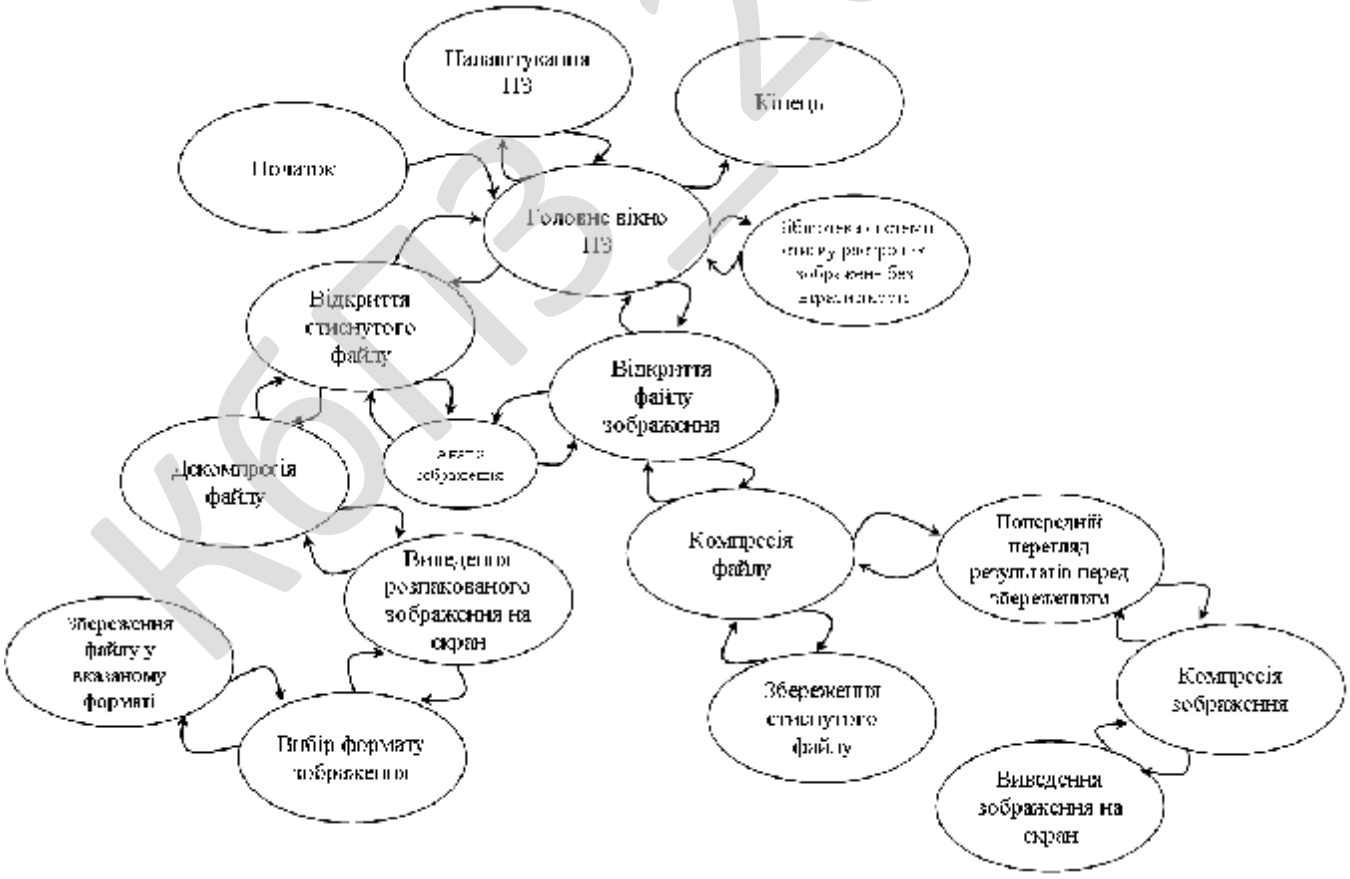

Рисунок 3.6 – Діаграма взаємодії

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків:

– Виведення головного вікна ПЗ.

– Запит відкриття зображення.

– Завантаження та формування контейнеру із зображенням.

– Виведення контейнеру на екран.

– Запит стиснення зображення.

– Вибір величини зміщення домену.

– Вибір величини розміру регіону.

– Підпрограма стиснення зображення.

– Виведення зображення та інформації стиснення.

– Збереження стиснутого зображення.

– Запит декомпресії.

– Підпрограма декомпресії зображення.

– Виведення зображення після декомпресії.

– Збереження декомпресійного зображення.

– Сигнал WM_CLOSE (запит).

На рисунку 4.2 наведено блок-схему підпрограми стиснення зображення. Її робота складається з виконання наступних кроків:

– Конвертація початкового зображення на сітку частин.

– Формування вихідного файлу.

– Встановлення кінцевого розміру вихідного файлу відносні сітки частин.

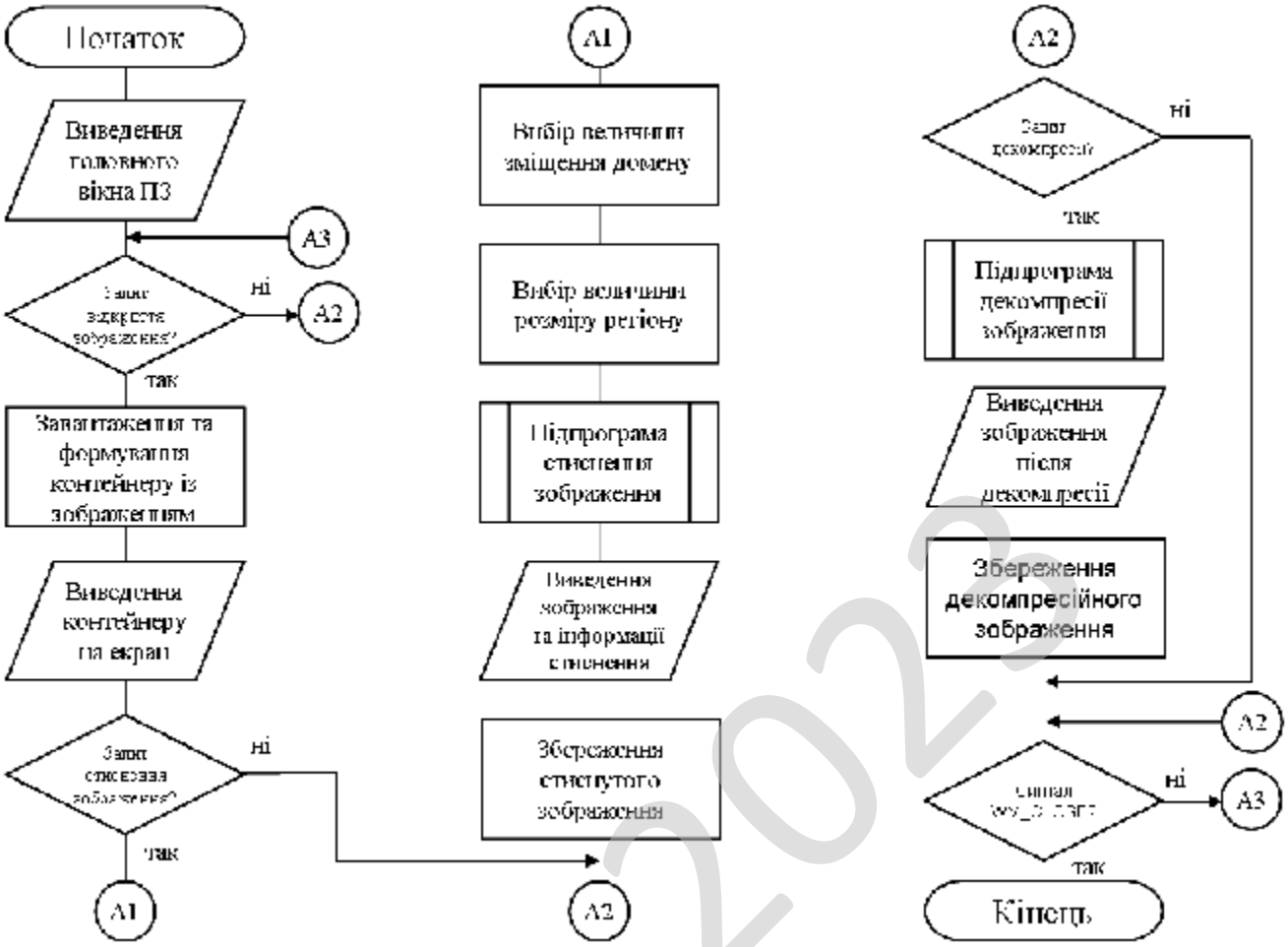

Рисунок 4.1 – Блок-схема основної програми

– Формування тимчасового контейнеру зображення.

– Заповнення тимчасового контейнеру початковим зображенням.

– Зменшення роздільної здатності зображення тимчасового контейнеру.

– Конвертація тимчасового контейнеру зображення на сітку частин.

– Перетворення частин.

– Заміна частин тимчасового контейнеру з частинами вихідного файлу.

– Запис у вихідний файл коефіцієнтів перетворень.

– Запис у файл різниці середньої яскравості між частинами.

– Закриття вихідного файлу.

Рисунок 4.2 – Блок-схема підпрограми стиснення зображення

Опис алгоритмів функціонування системи

Опишемо процедури, які реалізують алгоритм стиснення:

```
type
// Опис одного регіону у вихідному файлі становить всього-на-всього 6 байт
// У такий спосіб розмір файлу = Кількості регіонів * 6
        TIfsRec = packed record
          DomCoord, DomCoord: Word;
// Координати лівого верхнього кута домену
          Betta, FormNum: Byte;
// Розходження в яскравості, Номер перетворення
        end;
        TRegionRec = packed record
          MeanColor: Integer;
```

```
// Усереднена кольоро-яскравість
            Ifs: TIfsRec;
// Параметри, що обчислюються при компресії
          end;
          TDomainRec = packed record
            MeanColor: Integer;
// Усереднена кольорояскравість
          end;
// Заголовок файлу (8 байт)
          TIfsHeader = packed record
            FileExt: array[1..3] of Char;
            RegSize: Byte;
// Розмір регіону
            XRegions, YRegions: Word;
// Кількості регіонів по X и B
          end;
// Типи афіних перетворень
        TTransformType = (ttRot0, ttRot90, ttRot180, ttRot270, ttSimm, ttSimm,
ttSimmDiag1, ttSimmDiag2);
        TProgressProc = procedure(Percent: Integer; TimeRemain: Cardinal) of
Object;
          TFractal = class(TComponent)
          private
            SourImage: PByteArray;
// Пікселі зображення після перетворення в сірий
            DomainImage: PByteArray;
// Масив пікселів доменного зображення
            SourWidth: Integer;
// Ширина зображення
            SourHeight: Integer;
// Висота зображення
            FRegionSize: Integer;
// Розмір регіону
            FDomainOffset: Integer;
// Зсув доменів
            Regions: array of array of TRegionRec;
// Інформація про регіони
            Domains: array of array of TDomainRec;
// Інформація про домени
            FGamma: Real;
            FMaxImageSize: Integer;
```

```
// Максимально припустимий розмір зображення
            FStop: Boolean;
            FIfsIsLoad: Boolean;
// Перевіряє, чи була виконана компресія ( чизавантажені IFS-дані)
            FBaseRegionSize: Integer;
// Розмір регіону при стиску
// Очищає дані
            procedure ClearData;
// Генерує виняткову ситуація з повідомленням Msg
            procedure Error(Msg: string; Args: array of const);
// Створює масив посилань Regions на регіони
            procedure CreateRegions;
// По вихідному зображенню SourImage створює доменне зображення
            procedure CreateDomainImage;
// Створює масив 2-мірний Domains, у який заноситься усереднена кольорояскравість
             для кожного домену
            procedure CreateDomains;
// Визначає усереднену яскравість для ділянки Image з початком у т. (X, Y)
            function GetMeanBrigth(Image: PByteArray; X, Y, Width: Integer): Byte;
            function XRegions: Integer;
// Число регіонів по X
            function YRegions: Integer;
// Число регіонів по B
            function XDomains: Integer;
// Число доменів по X
            function YDomains: Integer;
// Число доменів по B
            function DomainImageWidth: Integer;
// Ширина доменного зображення
            procedure SetGamma(const Value: Real);
            procedure SetMaxImageSize(const Value: Integer);
            procedure SetRegionSize(const Value: Integer);
            procedure SetDomainOffset(const Value: Integer);
// Трансформує заданий регіон у відповідності з TransformType. Пікселі в
// заданому регіоні повинні йти один за одним
            procedure TransformRegion(Sour, Dest: PByteArray; TransformType:
TTransformType);
            {Повертає різницю (метрична відстань) між регіоном і доменом
            function GetDifference(Region: PByteArray; DomCoord, DomCoord, Betta:
Integer): Integer;
// Копіює зазначений регіон з масиву AllImage у масив Dest.
// Width – ширина масиву AllImage
```

```
            procedure CopyRegion(AllImage, Dest: PByteArray; X, Y, Width: Integer);
            function GetPixel(X, Y: Integer): Byte;
          public
            constructor Create(AOwner: TComponent); override;
            destructor Destroy; override;
// Виконує властиво сам стиск. При UseAllTransform будуть виконані
// всі афінні перетворення: поворот і симетрія. У противному випадку
// буде виконаний тільки поворот
            procedure Compress(UseAllTransform: Boolean = True; BackProc:
TProgressProc = nil);
// Примусово перериває процес фрактального стиску
            procedure Stop;
// Виконує розпакування зображення. IterCount – кількість ітерацій розпакування,
// RegSize – розмір регіону з розпакованому зображенні. Якщо ця величина
// така ж, як RegionSize при стиску, то розмір зображення буде як при стиску.
// При зменшенні RegSize розпаковане зображення зменшується й навпаки
            procedure Decompress(IterCount: Integer = 15; RegSize: Integer = 0);
// Будує зображення по доменам. Можна використовувати відразу
// після стиску для того, щоб перевірити якість стиску. Зображення,
// побудоване по доменам, схоже на відновлене зображення, тільки має більшу
// контрастність
            procedure BuildImageWithDomains;
// Перевіряє, чи була виконана компресія ( чизавантажені IFS-дані, необхідні
// для декомпресії). Якщо IfsIsLoad=True, то можна сміло робити декомпресію
            property IfsIsLoad: Boolean read FIfsIsLoad;
// Ширина зображення (вихідного, побудованого по доменам, або розпакованого)
            property ImageWidth: Integer read SourWidth;
// Висота зображення (вихідного, побудованого по доменам, або розпакованого)
            property ImageHeight: Integer read SourHeight;
// Повертає значення яскравості для зазначеного пікселя
            property Pixel[X, Y: Integer]: Byte read GetPixel;
// Завантажує повнокольорове зображення TBitMap для подальшої компресії
            procedure LoadImage(Image: TBitmap);
// Малює зображення на переданому TBitmap. При Regions = True рисується вихідне
// зображення, інакше рисується доменне зображення (воно таке ж, тільки
// в 4 рази менше по площі)
            procedure DrawImage(Image: TBitmap; Regions: Boolean = True);
// Зберігає результат стиску у двійковий файл
            procedure SaveToFile(FileName: string);
// Виконує завантаження даних із двійкового файлу
            procedure LoadFromFile(FileName: string);
```

```
// Визначає, який розмір буде в IFS-Файлу після компресії
          function GetIFSFileSize(): Cardinal;
        published
// Встановлює розмір регіону.
// Не можна змінювати розмір регіону
// після завантаження зображення для
// компресії, тому що завантажене
// зображення коректується в
// відповідності з RegionSize
          property RegionSize: Integer read FRegionSize write SetRegionSize;
// Величина зсуву домену. За замовчуванням = 1
// (це число відповідає доменному зображенню,
// що в 4 рази менше вихідного)
      property DomainOffset: Integer read FDomainOffset write SetDomainOffset;
// Колірний коефіцієнт Гама
          property Gamma: Real read FGamma write SetGamma;
// Максимальний розмір зображення
      property MaxImageSize: Integer read FMaxImageSize write SetMaxImageSize;
        end;
```

Наведемо частини коду, які реалізують деякі функції розробленого, у результаті виконання магістерського проектування, програмного продукту.

Завантаження зображення в об'єкт винятково з метою реалізації операції передперегляду:

```
          FractalComp.LoadImage(ImPreview.Picture.Bitmap);
          Edit1.Text := OpenPictureDialog1.FileName;
        except
          ImPreview.Picture := nil;
          ImPreview.Canvas.TextOut(5, 10, 'Помилка');
          ImPreview.Canvas.TextOut(5, 30, 'завантаження');
          lbSize.Caption := '';
          ImageExists := False;
```

Покажемо зображення на екрані повністю:

```
      procedure TForm1.Button2Click(Sender: TObject);
      begin
// Показує зображення на екрані повністю.
        if ImageExists then
        begin
          Application.CreateForm(TfmShowImage, fmShowImage);
          if CheckBox1.Checked then
```

```
        FractalComp.DrawImage(fmShowImage.Image1.Picture.Bitmap, False)
      else
        fmShowImage.Image1.Picture := ImPreview.Picture;
      with fmShowImage do
      begin
        AutoSize := True;
        Position := poScreenCenter;
        ShowModal;
      end;
      FreeAndNil(fmShowImage);
    end;
  end;
```

4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 9041:2020. Його повна назва: ДСТУ 9041:2020. Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса.

Цей алгоритм призначений для шифрування коротких (до 616 біт) повідомлень для будь-яких алгоритмів шифрування, в тому числі визначених національними стандартами України.

Як і стандарт цифрового підпису ДСТУ 4145:2002, новий алгоритм використовує криптографічні перетворення у групі точок еліптичних кривих, використовуючи замість кривих у формі Вейєрштрасса найновітніші розробки у галузі еліптичної криптографії – криві у формі Едвардса. Це дає суттєві переваги у швидкодії більш ніж у 3 рази.Новий стандарт розроблений з урахуванням усіх найсучасніших вимог до стійкості криптографічних алгоритмів. Так, нижня межа стійкості криптосистем у цьому стандарті дорівнює 2127 (≈1042) (це більш ніж у півтора рази вище, ніж у ДСТУ 4145) і можуть бути обрані інші рівні, такі як 2255 (≈1085), 2383 (≈10127) та 2767 (≈10255); крім того, строго обґрунтована його стійкість як до атак на відновлення відкритого тексту, так і до розрізнюючих атак.

Проект алгоритму шифрування, що ліг в основу цього стандарту, пройшов апробацію як в Україні, так і за її межами (Центрально-Європейська конференція з криптографії (червень 2020 року) – форум ведучих криптологів з усього світу).

Стандарт ДСТУ-9041 узгоджений з усіма діючими в Україні національними стандартами. Новиною стандарту є його сфера застосування – інкапсуляція ключів, найсучасніший математичний апарат, а також новий алгоритм генерації псевдовипадкових послідовностей, який, на відміну від аналогічного алгоритму генерації з ДСТУ 4145, використовує виключно національні криптографічні алгоритми національних стандартів та не містить посилань на відповідні пост-радянські стандарти, термін дії яких вже практично вичерпався.

Новий стандарт не належить до так званих постквантових стандартів. Але його стійкість буде під загрозою лише тоді, коли з'являться квантові комп'ютери з 700 і більше кубітами (на даний час кількість "робочих" кубітів, які вдалося створити, – близько 50). Його перевагою перед постквантовими алгоритмами є відносно невелика довжина ключа (у десятки або навіть у сотні разів менша, ніж у постквантових алгоритмах).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

1. Меню: Файл; Конвертор; Додатки; Налаштування; Довідка.
2. Блоків обрання папки та файлу.
3. Блоків вхідне та вихідне зображення.
4. Параметри компресії без втрати якості вхідного зображення: розмір файлу; Роздільна здатність; Встановлення значення частин та розміру частин.

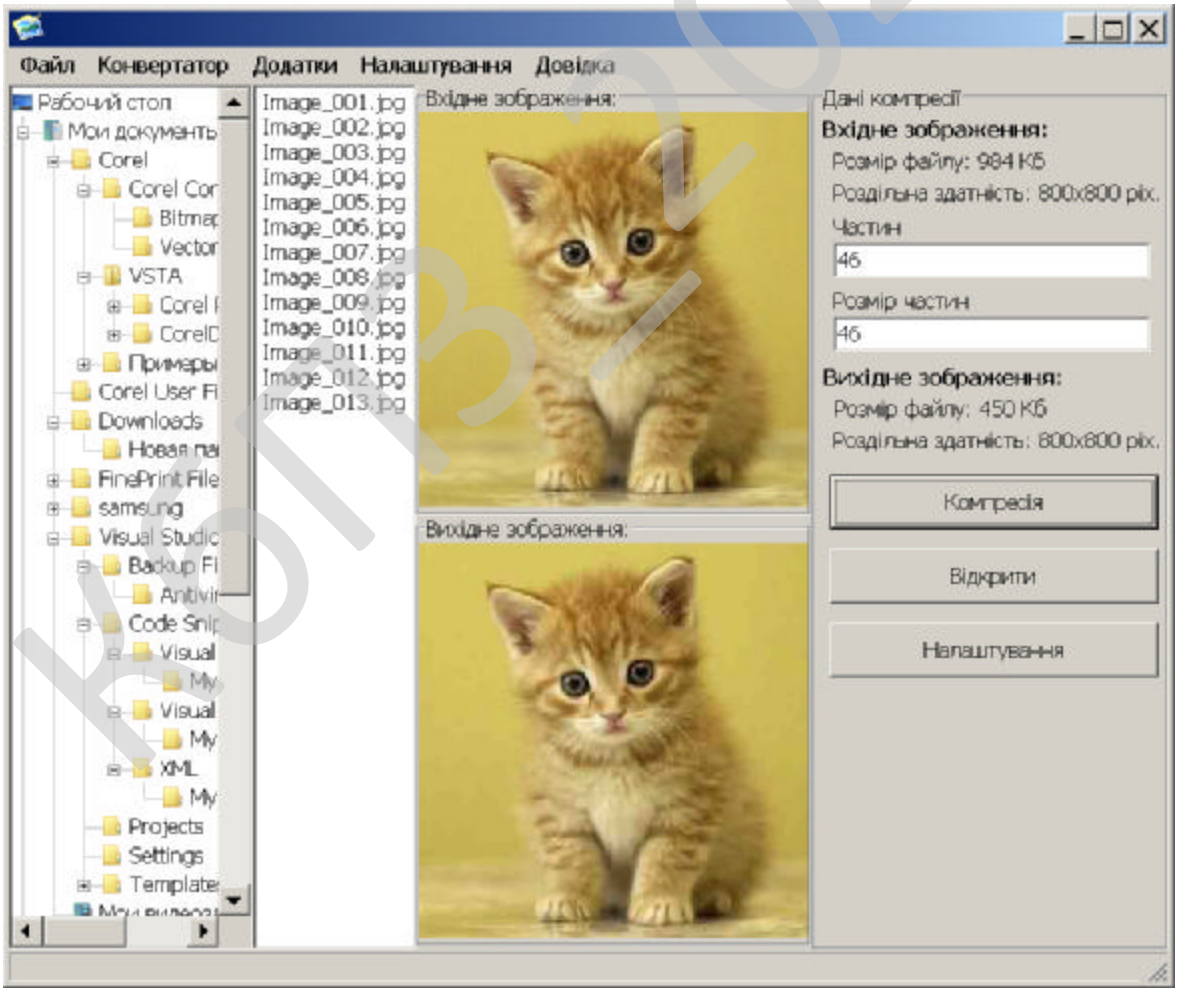

Рисунок 5.1 – Головне вікно програми

При розробці ПЗ була створена форма авторського права з ліцензій на використання програмного забезпечення.

Ліцензія на використання програмного забезпечення це вид ліцензії, що визначає умови використання виробу, яким є комп'ютерне програмне забезпечення.

Ліцензія може надавати дозвіл робити з ним речі, які були б інакше заборонені законом про авторське право. Наприклад, ліцензія на використання програмного забезпечення може дати дозвіл робити копії програмного забезпечення. Власник авторського права може запропонувати ліцензію на використання ПЗ односторонньо, або як частину ліцензійної угоди на використання ПЗ з іншою стороною.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи стиснення растрових зображень без втрати якості.

Метою розробки є дослідження та програмна реалізація системи стиснення растрових зображень без втрати якості.

Об'єктом дослідження є процес стиснення растрових зображень без втрати якості.

Предметом дослідження є методи стиснення растрових зображень без втрати якості.

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 растрових зображень без втрати якості.

– Розроблено вітчизняний продукт стиснення растрових зображень без втрати якості, 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 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи стиснення растрових зображень без втрати якості.

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	40
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

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

Продовження табл. 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)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	40000
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	55
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.:

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

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.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

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 $ПВj$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

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:

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

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot59=99 \text{ люд/день}$$

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	9	Д7
Робочий проект	99	Ф 7.1-7.4
Впровадження	13	Д13
Всього	140	–

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 Fpq – плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{140 \cdot 1}{60\text{-}5} = 2{,}55 \text{ ставки}$$

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 обладнання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м.п.	2,5	100	250	4
Кабельне господарство електромереж	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{227 \cdot 3}{1,2} = 567,5 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=567,5/(60·8)=1,2 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	Кількість штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, серверу доступу АДСЛ (ОС Linux), Wi-Fi налаштування ADSL, VPN, PPPoE, Frame Relay	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 д мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
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	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	1	11980	35940
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	2,55	13000	99450
Інженер-електронщик	1,2	11000	39600
Інженер-системотехнік	0,25	12000	9000
Адміністратор мережі	0,5	11000	16500
Системний програміст	0,25	11000	8250
Дизайнер WEB	0,25	11000	8250
Інженер-верстальник	0,25	11000	8250
Бухгалтер-економіст	0,5	11000	16500
Всього за період розробки	$R_{сп}$=7	-	$Ф_{роб}$=250740

Розрахуємо середньоденну зарплату одного виконавця:

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

де Фроб – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{250740}{7 \cdot 60} = 597 \; грн.$$

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 8 робочих місць. Sу – питома площа на одне робоче місце, $м^2$; Цпл – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пектораль» 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:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 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 Цм – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

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 Інтернет магазину Компбест за 28.10.23 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		10947
Системний блок		7347
Процесор	AMD Ryzen 5 4500 (100-000000644) AM4, 6 ядер, 12 потоків, 3.6 GHz, 4.1 GHz	-
Системна плата	GIGABYTE A520M H DDR4, 64 ГБ, 5100 MHz, LAN – 1 Гбіт/с, DVI, HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	Radeon R7 350 2Gb Afox (AFR7350 2048D5H4-V3) PCI-Express 3.0, 2 ГБ GDDR5, 128 Bit	-
Жорсткий диск	SSD M.2 2280 240GB Apacer 240 GB, TLC, M.2, SATA III (6Gb/s)	-
Оперативна пам'ять	DDR4 8GB 2666 MHz Kingston	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	-
Корпус	Vinga Apache-500W, Miditower, ATX, Micro – ATX, Mini – ITX, PSU – 500 Вт, Слотів розширення – 7, Front 2x12 см, Back 1x12 см, Side 2x12 см, GPU – до 370 мм, CPU – 166 мм	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	-
інше	Клавіатура, мишка	-
Система охолодження ЦП	Vinga Q6	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	Samsung S22R350FHI (LS22R350FHIXCI) / 21.5" (1920x1080) IPS LED / HDMI, VGA	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струменевий	Epson Stylus Photo P50 (C11CA45341) + USI cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

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	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

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	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	40000	10	4000
Разом	$К_р$= 1596075		$А_р$= 140140

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців

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

де Ne – Кількість екземплярів програм, шт.

$$З_о = 597 \cdot 140/40 = 2090 \text{ грн}$$

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, %

$$З_д = 2090 \cdot 10 \cdot 0{,}01 = 209 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=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, %

$$С_{оц} = 0{,}01 \cdot 22(2090+209) = 506 \text{ грн}$$

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

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

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 2090 \cdot 15 \cdot 0{,}01 = 314 \text{ грн}$$

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, картріджей, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно норм $n_{міс}$ приймаємо 1/6 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки $N_м$ =3 міс:

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

$$З_{М1} = 210 \cdot 3 \cdot 1/6 = 105 \text{ грн}.$$

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків в кількості 10 примірників:

$$З_{М2} = \sum Ц_{д}., \quad (7.17)$$

де Цд – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 49,2 грн/шт., DVD-R LG 4,7Gb, 16x speed Cake box – 49,2 грн/шт.

$$З_{М2} = 49{,}2 \cdot 10 = 492 \text{ грн.}$$

Згідно виданих 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.

$$З_{М3} = 574+558+570=1702 \text{ грн.}$$

$$З_{М} = (105 + 492 + 1702)/40=57 \text{ грн.}$$

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

$$О_{п} = З_{о} \cdot Н_{п} \cdot 0{,}01, \quad (7.19)$$

де Нп – норматив витрат на освоєння нових мов програмування, %

$$О_{п} = 2090 \cdot 15 \cdot 0{,}01=314 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 40 прим.)

$$А_{м} = \frac{А_{р} \cdot N_{міс}}{N_{е} \cdot 12}, \quad (7.20)$$

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_{м} = 140140 \cdot 3/(40 \cdot 12)=876 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$С_{п} = З_{о} + З_{д} + С_{оц} + Г_{осп} + З_{м} + О_{п} + А_{м}. \quad (7.21)$$

$$C_n = 2090+209+506+314+57+314+876 = 4366 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 55%

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де Рс – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 4366 = 2401 \text{ грн.}$$

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. Основна зарплата виконавців	$З_о$	2090
2. Додаткова зарплата виконавців	$З_д$	209
3. Відрахування на соціальні потреби	$С_{оц}$	506
4. Загальногосподарські витрати	$Г_{осп}$	314
5. Витрати на матеріали	$З_М$	57
6. Освоєння нових операційних систем, мов програмування	$О_п$	314
7. Амортизація основних фондів	$А_м$	876
8. Повна собівартість програмного забезпечення	$С_п$	4366
9. Плановий прибуток	$П_р$	2401
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	6767
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1353,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	9168

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	–	9168
Всього капітальних витрат	–	9168

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 технічне обслуговування	$З_р$	17178	4294
2. Витрати на електроенергію	$З_{ел}$	12996	12455
3. Витрати на амортизацію	$З_{ам}$	0	4584
Всього витрат за рік	$І$	30174	21333

Витрати на оплату праці:

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

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 80 годин на рік до 20 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\,баз} = 80 \cdot 160 \cdot 1{,}1 \cdot 1{,}22 = 17178 \text{ грн.}$$

до

$$З_{р\,нов} = 20 \cdot 160 \cdot 1{,}1 \cdot 1{,}22 = 4294 \text{ грн.}$$

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

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

Зел баз=·0,475·7200·3,8 = 12996 грн

Зел нов=·0,475·6900·3,8 = 12455 грн

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 продукція	50	–	9168	–	4584
Всього відрахувань	-	–	9168	–	4584

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

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

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

$$Ев = (6767 - 4366) \cdot 40 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 40000) \cdot 3/12 = 61005 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

де: $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{188075}{(6767 - 4366) \cdot 40 \cdot 12 / 3} = 0,5 \; років.$$

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, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (30174 - 21333) - 0,5 \cdot 9168 = 4257 \text{ грн.}$$

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

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

$$T_{сп} = \frac{9168}{30174 - 21333} = 1 \; рік$$

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.	40
2. Повна собівартість розробленої програми	Грн	4366
3. Ціна розробленої програми	Грн.	6767
4. Плановий прибуток від реалізації розробленої програми	Грн.	2401
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1596075
7. Загальний прибуток від реалізації програмної продукції	Грн.	96040
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	61005
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	9168
11.Величина економічного ефекту у користувача програмної продукції	Грн.	4257
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	1

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

Програмісти у процесі роботи отримують негативний вплив на органи зору, а також мають значну розумову напругу і нервово-емоційне навантаження. Руки (м’язи рук та суглоби пальців) при роботі з клавіатурою мають теж істотне навантаження. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій спеціалісти відносять високочастотні електромагнітні коливання роботи апаратної частини ЕОМ та виділення шкідливих газів. Ці шкідливі фактори можуть привести до професійних захворювань. До недоліків умов праці користувачів комп’ютерної техніки можна віднести:

– недостатню площу і обсяг виробничого приміщення;

– недотримання вимог, мікроклімату на робочих місцях;

– низький рівень освітленості у приміщеннях і на робочих поверхнях апаратури;

– підвищений рівень низькочастотних магнітних полів від моніторів;

– порушення вимог організації робочих місць;

– недотримання вимог до режимам праці та відпочинку;

– надмірне виробничу навантаження працівників;

– відсутність навичок зниження впливу психоемоційного напруги.

Відповідно до ст.14 Закону «Про охорони праці» [3] на роботодавця покладено обов'язок забезпечити: безпеку працівників при експлуатації устаткування; застосування коштів індивідуальної захисту працівників; відповідні вимоги охорони праці, умови праці в кожному робоче місце; дотримання режиму праці та відпочинку працівників; навчання безпечним методам і прийомам виконання; інструктаж з охорони праці; організацію контролю над станом умов праці в робочих місць; проведення атестації робочих місць в умовах праці.

Максимально зменшити кількість шкідливих впливів на людину при високій продуктивності праці, створити комфортні умови для роботи людей – ось одна з головних задач охорони праці [5].

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальни машини (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, параметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98 [2].

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

Ці фактори можуть викликати у працівника певні розлади здоров'я, зокрема підвищення артеріального тиску, кон'юктивіти, тендовагініти ті інші захворювання.

Комп'ютер, як і будь-який електричний прилад, особливо при його неправильному підключенні, може бути джерелом ураження оператора електричним струмом. Саме тому всі працівники, які працюють з персональним комп'ютером, повинні мати першу(або другу) групу допуску з електробезпеки.

Через наявність зазначених факторів працівники, які працюють з персональними комп'ютерами, підлягають попередньому та періодичному медичному огляду згідно з пунктом 6.2.3 додатку 4 до наказу Міністерства охорони здоров'я України "Про затвердження Порядку проведення медичних оглядів працівників певних категорій" від 21 травня 2007 року №246 [8].

8.3 Аналіз умов праці на робочому місці фахівця

Робота програміста пов'язана з постійною роботою на ЕОМ, яка відбувається у кімнаті розмірами 4,4 м×6,2 м×2,9 м. Одна з її більших стін має шість двостулкових вікон, розмірами 2 м×1,8 м, які виходять на північний захід. Вікна розташовані рівномірно по всій довжині стіни. Підлога в кімнаті покрита леноліумом, всі стіни пофарбовані світло оранжевого кольору до висоти 2,8 м, а далі підвісна стеля. Уздовж стін розташовані комп'ютерні столи. На них розташовуються 2 персональні комп'ютери й інша оргтехніка (сканер принтери, телефони й ксерокс). Столи мають пластикове покриття. Габарити їхньої робочої поверхні 1255 мм×845 мм. Висота столів 760 мм. Висота стільців від рівня підлоги становить 430 мм.

Згідно НПАОП 0.00 – 1.28 – 10 «Правила охорони праці під час електронно-обчислювальних машин» площа повинна задовольняти умові – не менш 6 м^2 на одне робоче місце. Кратність повітрообміну в приміщенні вузла також регламентується ДСанПіН 3.3.2.007-98 [2], вона повинна становити 20 м^3/годину на одне місце. Виконання даних вимог забезпечить підтримку в приміщенні вузла оптимального значення вологості й складу повітря.

Відповідно ДБН В.2.5 – 28 – 2006 [1] роботу програміста 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 вузла 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 сполученому висвітленні), повинен становити 0,5%, освітленість при штучному висвітленні повинна становити 300 лк.

За результатами виміру освітленості відділом охорони праці величина освітленості від системи загального штучного висвітлення лежить у межах 200-250 лк, що не відповідає вимогам, які пред'являються до приміщення.

Відповідно ДСанПіН 3.3.2.007-98 [2] рівні звукового тиску в робочому приміщенні не повинні перевищувати в октавних смугах із середньо геометричними частотами наступних значень, наведених у таблиці 8.1.

Таблиця 8.1 – Допустимі спектри рівнів звукового тиску

Робоче місце	Рівень звукового тиску, дБ, в октавних смугах із середньогеометричними частотами, Гц								Рівень звуку і еквівалентний рівень звуку, дБА
	63	125	250	500	1000	2000	4000	8000	
Приміщення конструкторських бюро, програмістів обчислювальних машин, лабораторій для теоретичних робіт і опрацювання експериментальних даних, прийому хворих в медпунктах	71	61	54	49	45	42	40	38	50

У приміщенні перебувають наступні джерела шуму: електродвигуни внутрішнього вентилятора ЕОМ; працюючі принтери; працюючі дисководи. Шум, вироблений вентилятором можна класифікувати як постійний, всі інші джерела шуму, як імпульсні. Відповідно паспорта на приміщення рівень звуку, Дб(А), обмірюваний за шкалою (А) шумоміра досяг величини 28,3 Дб(А) при роботі всього встаткування вузла, включаючи й ксерокс. Це дозволяє зробити висновок

про відповідність рівня звуку в приміщенні вимогам нормативних актів. Ергономічні вимоги до робочого місця працюючого з ВДТ ЕОМ і ПЕОМ нормуються НПАОП 0.00 – 1.28 – 10. Оптимальне положення тіла того, що працює забезпечується відповідною конструкцією робочого місця, а також регуляцією висоти робочої поверхні, сидіння, простори й підставки для ніг. Даного місця програміста не мають регульованих параметрів. Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту дані в таблиці 8.2.

Таблиця 8.2 – Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту

Ріст людини, см	Висота робочої поверхні мм,	Висота простору для ніг, мм	Висота робочого сидіння, мм
175	765(740)	655(600)	450(440)

У дужках зазначені реальні значення параметрів робочого місця; всі вони не відповідають параметрам, зазначеним у стандарті.

Параметри мікроклімату можуть мінятися в широких межах, тоді як необхідною умовою життєдіяльності людини є підтримка сталості температури тіла завдяки властивості терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах ДСН 3.3.6.042 – 99 [4] встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. табл. 8.3).

Таблиця 8.3 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні	22 – 24°С
	Відносна вологість	40 – 60%
	Швидкість руху повітря	до 0,1 м/с
Теплий	Температура повітря в приміщенні	23 – 25°С
	Відносна вологість	40 ... 60%
	Швидкість руху повітря	0,1 ... 0,2 м / с

8.4 Розробка заходів з умов поліпшення охорони праці

Провівши аналіз умов праці в розглянутому вище приміщенні, було отримано наступні результати:

– значення мікроклімату в приміщенні не перевищує норму;

– розрахунки розміру робочого місця на одного працівника відповідають нормі;

– рівень шуму в приміщенні не становить вище норми.

З вище перечислених результатів можна зробити висновок, що основний вплив на продуктивність ІТ-спеціалістів є його психологічний стан. Тому є доцільним зменшити рівень стресу на робочому місці.

Рекомендовані наступні заходи: За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів необхідно відділяти одне від одного перегородками висотою 1,5 – 2м. Конструкція робочого місця користувача персонального комп’ютера має забезпечити підтримання оптимальної робочої пози офісного працівника. Конструкція робочого столу має відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання (дисплея, клавіатури, принтера) і документів. Висота робочої поверхні робочого столу має

регулюватися в межах 680-800 мм, а ширина і глибина – забезпечувати можливість виконання операцій у зоні досяжності моторного поля (рекомендовані розміри: 600-1400мм, глибина – 800-1000мм). Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600 мм, завширшки не менше ніж 500 мм, завглибшки (на рівні колін) не менше ніж 450 мм, на рівні простягнутої ноги не менше ніж 650 мм. Робочий стілець має бути підйомно-поворотним, регульованим за висотою, з кутом і нахилу сидіння та спинки і за відстанню від спинки до переднього краю сидіння поверхня сидіння має бути плоскою, передній край – заокругленим [7].

8.5 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток *80·80·8 мм*., (згідно з ДСТУ 2251:2018 «Кутики сталеві гарячекатані рівнополичні. Сортамент») довжиною *L=1,6 м.,* та горизонтальний електрод – металева полоса з перетином *60·5 мм*. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проведем за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *H=0,5 м*. Відстань між вертикальними заземлювачами (електродами) *A=1,6 м.* Глибина закладення горизонтального контура заземлення *t=0,6 м*. Опір заземлювача, який нормується: $R_{3H} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,6+1,6/2=1,4 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi\rho = 1{,}36 \cdot 40 = 54{,}5 \text{ Ом·м.}$$

де $\psi = 1{,}36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [6];

$\rho_2 = 40$ Ом·м. – табличне значення питомого опору нижнього шару грунта (глина) [7].

Еквівалентний діаметр вертикального електрода (кутка) [7]:

$$Dв = 0{,}95 \cdot К = 0{,}95 \cdot 80 = 76{,}85 \text{ мм.} = 0{,}076 \text{ м.}$$

де $К = 80$ мм. – розмір полиці метелевого кутка.

Відношення $A/L = 1{,}6/1{,}6 = 1$

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [7]:

$$R_O = 0{,}366(\rho/L)[\lg(2L/Dв) + (1/2)\lg((4T+L)/(4T-L))] =$$

$$= 0{,}366(54{,}5/1{,}6)[(\lg(2 \cdot 1{,}6/0{,}076) + (1/2)\lg((4 \cdot 1{,}4 + 1{,}6)/(4 \cdot 1{,}4 - 1{,}6))] =$$

$$= 21{,}8 \text{ Ом.}$$

Визначаємо коефіцієнт екранування вертикальних електродів $Кев = 0{,}8$ при орієтовній кількості вертикальних електродів, яке дорівнює 5 [7].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без врахування горизонтального заземлювача), при $R_{ЗН} = 4$ Ом:

$$N = R_O / (Кев \, R_{ЗН}) = 21{,}8 / (0{,}8 \cdot 4) = 6{,}8 \approx 7 \text{ шт.}$$

Визначаємо довжину з'єднуючої полоси:

$$L_П = 1{,}05 \cdot A \cdot N = 1{,}05 \cdot 1{,}6 \cdot 7 = 12{,}2 \approx 12 \text{ м.}$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [7]:

$$R_П = 0{,}366(\rho \cdot K_П / L_П)\lg(2 \cdot L_П^2 / (B \cdot t)) =$$

$$= 0{,}366(40 \cdot 5/12) \cdot \lg((2 \cdot 12^2)/(0{,}06 \cdot 0{,}6)) = 20{,}16 \text{ Ом.}$$

де $K_П = 5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичной зони для з'єднуючої полоси [7]:

$B = 60$ мм. $= 0{,}06$ м.- ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [7]:

$$R = (R_O \cdot R_П)/(R_O \cdot \eta_П + N \cdot R_П \cdot Кев) =$$

$$= (21,8 \cdot 20,16)/(21,8 \cdot 0,75 + 9 \cdot 20,16 \cdot 0,8) = 3,46 \text{ Ом.}$$

де $\eta_П = 0,75$ – табличне значення коефіцієнта екранування з'єднуючої полоси [7].

Умова $R \leq R_{ЗН}$ виконується ($3,46 \leq 4$).

Так як при *7* вертикальних електродах *R* суттєво менше $R_{ЗН}$, зменшимо кількість вертикальних електродів *N* до *6* і виконаємо перерахунок. У результаті остаточно отримали: $R = 3,96$ *Ом.* при кількості вертикальних електродів $N=6$.

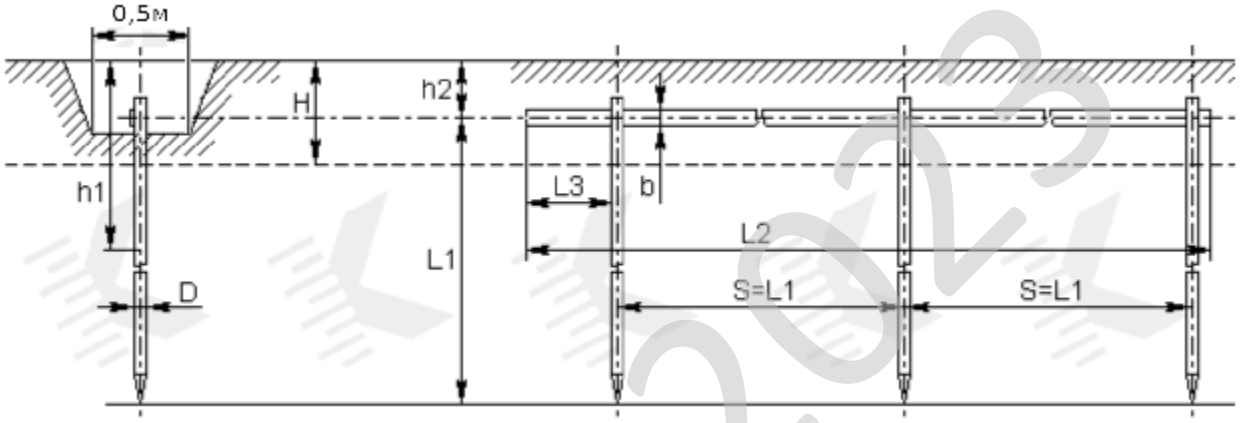

Рисунок 8.1 – Схема штучного заземлення.

Висновки до розділу

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

З цих міркувань було здійснено аналіз умов праці на робочому місці програміста, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

Список використаних джерел інформації

1. Державні будівельні норми України ДБН В.2.5 – 28 – 2006: Інженерне обладнання будинків і споруд. Природне і штучне освітлення. – *Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0168667-06#Text* (дата звернення 19.10.22).

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 (дата звернення 19.10.22).

3. 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 (дата звернення 19.10.22).

4. 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* (дата звернення 19.09.22).

5. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. – 19 с. [Електронний ресурс]. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12240 (дата звернення 19.09.22).

6. Охорона праці. Ч. 1. Захисне заземлення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград : КІСМ, 1997. – 20 с. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358 (дата звернення 19.09.22).

7. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : "Колос", 1973. – 238 с.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи стиснення растрових зображень без втрати якості.

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 растрових зображень без втрати якості.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем стиснення растрових зображень без втрати якості.

– Досліджена система стиснення растрових зображень без втрати якості.

– На основі отриманих результатів досліджень створена програмна реалізація системи стиснення растрових зображень без втрати якості.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання стиснення растрових зображень без втрати якості.

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 Delphi 10.4. 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 ДСТУ 9041:2020.

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 4257 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мікіньов В.І. Дослідження та програмна реалізація системи стиснення растрових зображень без втрати якості // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. M.F. Barnsley, Fractals Everywhere. London: Academic Press Inc., 1988.
3. A.E. Jacquin, “Fractal image coding based on a theory of iterated contractive image transformations,” in Proceedings of SPIE Visual Communications and Image Processing ’90, vol. 1360, pp. 227-239 1990.
4. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.
5. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.
6. 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.
7. 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.
8. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph.. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.
9. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.
10. 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.

11. 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,

12. 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.

13. 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

14. 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.

15. 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.

16. 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.

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

18. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and

cybersecurity of virtual cloud resources». Journal of theoretical and applied information technology Vol.98. No 21, 2020, P. 3334-3346.

19. Smirnov O., Kuznetsov A., 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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).

43. 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

44. 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.

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

46. 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.

47. 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.

48. 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

49. 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.

50. Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. Коваленка О.В., Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки України від 26.02.2013 року № 1/11-4368. – Кіровоград: КНТУ 2013. – 257с.

51. 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.

52. Смірнов О.А., Доренський О.П., Дрєєв О.М. Аналіз процесів стиснення та відновлення зображень на основі цифрових методів. Наука і техніка Повітряних сил Збройних Сил України. – Випуск 3(12). – Х.: ХУПС. – 2013. – С.122-127.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

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

					ВКРМ-122.23.0064.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Мікіньов В.І.			Дослідження та програмна реалізація системи стиснення растрових зображень без втрати якості	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Буравченко К.О.				М	
Н. Контр.		Коваленко А.С.				ЦНТУ КН-22М-2	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи стиснення растрових зображень без втрати якості.

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. № 33-13 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 растрових зображень без втрати якості.

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 растрових зображень без втрати якості;

– 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 Delphi 10.4.

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 18.12.2023 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

_____ Буравченко К.О.

Дослідження та програмна реалізація системи стиснення растрових зображень без втрати якості

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2023 року

FractalCompression.pas - файл алгориму стиску растрових зображень без втрати якості за допомогою фракталів

```
unit FractalCompression;

interface

uses
  Windows, Messages, SysUtils, Graphics, Classes;

type
  // Опис одного регіону у вихідному файлі становить всього-на-всього 6 байт
  // У такий спосіб розмір файлу = Кількості регіонів * 6
  TIfsRec = packed record
    DomCoord, DomCoord: Word; // Координати лівого верхнього кута домену
    Betta, FormNum: Byte; // Розходження в яскравості, Номер перетворення
  end;

  TRegionRec = packed record
    MeanColor: Integer; // Усереднена кольорояскравість
    Ifs: TIfsRec; // Параметри, що обчислюються при компресії
  end;

  TDomainRec = packed record
    MeanColor: Integer; // Усереднена кольорояскравість
  end;

  // Заголовок файлу (8 байт)
  TIfsHeader = packed record
    FileExt: array[1..3] of Char;
    RegSize: Byte; // Розмір регіону
    XRegions, YRegions: Word; // Кількості регіонів по X и B
  end;

  // Типи афіних перетворень
  TTransformType = (ttRot0, ttRot90, ttRot180, ttRot270, ttSimm, ttSimm,
ttSimmDiag1, ttSimmDiag2);

  TProgressProc = procedure(Percent: Integer; TimeRemain: Cardinal) of Object;

  TFractal = class(TComponent)
  private
    SourImage: PByteArray;  // Пікселі зображення після перетворення в сірий
    DomainImage: PByteArray;// Масив пікселів доменного зображення
    SourWidth: Integer;     // Ширина зображення
    SourHeight: Integer;    // Висота зображення
    FRegionSize: Integer;   // Розмір регіону
    FDomainOffset: Integer; // Зсув доменів
    Regions: array of array of TRegionRec; // Інформація про регіони
    Domains: array of array of TDomainRec; // Інформація про домени
    FGamma: Real;
    FMaxImageSize: Integer; // Максимально припустимий розмір зображення
    FStop: Boolean;
    FIfsIsLoad: Boolean; // Перевіряє, чи була виконана компресія (
чизавантажені IFS-дані)
    FBaseRegionSize: Integer;  // Розмір регіону при стиску

    {Очищає дані}
    procedure ClearData;

    {Генерує виняткову ситуація з повідомленням Msg}
    procedure Error(Msg: string; Args: array of const);

    {Створює масив посилань Regions на регіони }
    procedure CreateRegions;
```

```
    {По вихідному зображенню SourImage створює доменне зображення}
    procedure CreateDomainImage;

    {Створює масив 2-мірний Domains, у який заноситься усереднена
кольороякравість
     для кожного домену}
    procedure CreateDomains;

    {Визначає усереднену яскравість для ділянки Image з початком у т. (X, Y)}
    function GetMeanBrigth(Image: PByteArray; X, Y, Width: Integer): Byte;

    function XRegions: Integer; // Число регіонів по X
    function YRegions: Integer; // Число регіонів по B

    function XDomains: Integer; // Число доменів по X
    function YDomains: Integer; // Число доменів по B
    function DomainImageWidth: Integer; // Ширина доменного зображення

    procedure SetGamma(const Value: Real);
    procedure SetMaxImageSize(const Value: Integer);

    procedure SetRegionSize(const Value: Integer);
    procedure SetDomainOffset(const Value: Integer);

    {Трансформує заданий регіон у відповідності з TransformType. Пікселі в
     заданому регіоні повинні йти один за одним}
    procedure TransformRegion(Sour, Dest: PByteArray; TransformType:
TTransformType);

    {Повертає різницю (метрична відстань) між регіоном і доменом}
    function GetDifference(Region: PByteArray; DomCoord, DomCoord, Betta:
Integer): Integer;

    {Копіює зазначений регіон з масиву AllImage у масив Dest.
     Width - ширина масиву AllImage}
    procedure CopyRegion(AllImage, Dest: PByteArray; X, Y, Width: Integer);
    function GetPixel(X, Y: Integer): Byte;
  public
    constructor Create(AOwner: TComponent); override;

    destructor Destroy; override;

    {Виконує властиво сам стиск. При UseAllTransform будуть виконані
     всі афінні перетворення: поворот і симетрія. У противному випадку
     буде виконаний тільки поворот}
    procedure Compress(UseAllTransform: Boolean = True; BackProc: TProgressProc
= nil);

    {Примусово перериває процес фрактального стиску}
    procedure Stop;

    {Виконує розпакування зображення. IterCount - кількість ітерацій
розпакування,
     RegSize - розмір регіону з розпакованому зображенні. Якщо ця величина
     така ж, як RegionSize при стиску, то розмір зображення буде як при стиску.
     При зменшенні RegSize розпаковане зображення зменшується й навпаки}
    procedure Decompress(IterCount: Integer = 15; RegSize: Integer = 0);

    {Будує зображення по доменам. Можна використовувати відразу після стиску для
того,
     щоб перевірити якість стиску. Зображення, побудоване по доменам,
     схоже на відновлене зображення, тільки має більшу контрастність}
    procedure BuildImageWithDomains;

    {Перевіряє, чи була виконана компресія ( чизавантажені IFS-дані, необхідні
     для декомпресії). Якщо IfsIsLoad=True, то можна сміло робити декомпресію}
    property IfsIsLoad: Boolean read FIfsIsLoad;

    {Ширина зображення (вихідного, побудованого по доменам, або розпакованого)}
```

```
    property ImageWidth: Integer read SourWidth;

    {Висота зображення (вихідного, побудованого по доменам, або розпакованого)}
    property ImageHeight: Integer read SourHeight;

    {Повертає значення яскравості для зазначеного пікселя}
    property Pixel[X, Y: Integer]: Byte read GetPixel;

    {Завантажує повнокольорове зображення TBitMap для подальшої компресії}
    procedure LoadImage(Image: TBitmap);

    {Малює зображення на переданому TBitmap. При Regions = True рисується
вихідне
     зображення, інакше рисується доменне зображення (воно таке ж, тільки
     в 4 рази менше по площі)}
    procedure DrawImage(Image: TBitmap; Regions: Boolean = True);

    {Зберігає результат стиску у двійковий файл}
    procedure SaveToFile(FileName: string);

    {Виконує завантаження даних із двійкового файлу}
    procedure LoadFromFile(FileName: string);

    {Визначає, який розмір буде в IFS-Файлу після компресії}
    function GetIFSFileSize(): Cardinal;
  published
    {Установлює розмір регіону.
     УВАГА! Не можна змінювати розмір регіону після завантаження зображення для
     компресії, тому що завантажене зображення коректується в
     відповідності з RegionSize}
    property RegionSize: Integer read FRegionSize write SetRegionSize;

    {Величина зсуву домену. За замовчуванням = 1 (це число відповідає
     доменному зображенню, що в 4 рази менше вихідного)}
    property DomainOffset: Integer read FDomainOffset write SetDomainOffset;

    {Колірний коефіцієнт Гама}
    property Gamma: Real read FGamma write SetGamma;

    {Максимальний розмір зображення}
    property MaxImageSize: Integer read FMaxImageSize write SetMaxImageSize;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Samples', [TFractal]);
end;

{ TFractal }

procedure TFractal.ClearData;
begin
  if Assigned(SourImage) then FreeMem(SourImage);
  if Assigned(DomainImage) then FreeMem(DomainImage);
  SourImage := nil;
  DomainImage := nil;
  SourWidth := 0;
  SourHeight := 0;
  Regions := nil;
  Domains := nil;
end;

procedure TFractal.Compress(UseAllTransform: Boolean = True; BackProc:
TProgressProc = nil);
var
```

```
  Reg, Reg, Dom, Dom, Error, BestError, Betta, TransNum, TransCount: Integer;
  Region, BaseRegion: PByteArray;
  DCoord, DCoord, BestFormNum, BestDom, BestDom, BestBetta: Integer;
  Percent: Real;
  Tc, OneRegTime, AllRegTime: Cardinal;
label
  LExit;
begin
  FStop := False;
  FIfsIsLoad := False;

  FBaseRegionSize := RegionSize;

  if UseAllTransform then TransCount := 8 else TransCount := 4;

  if SourImage = nil then
    raise Exception.Create('Зображення для фрактального стиску ще не
завантажено!');

  CreateRegions;
  CreateDomains;

  GetMem(BaseRegion, RegionSize * RegionSize);
  GetMem(Region, RegionSize * RegionSize);


  OneRegTime := 0;
  AllRegTime := 0;
  Tc := GetTickCount;
  // Перебираємо регіони
  for Reg := 0 to YRegions - 1 do
    for Reg := 0 to XRegions - 1 do
    begin
      if Reg * XRegions + Reg > 10 then Tc := GetTickCount;

      Percent := (Reg * XRegions + Reg) / (YRegions * XRegions) * 100;
      BestError := $7FFFFFFF;
      BestFormNum := -1;
      BestDom := -1;
      BestDom := -1;
      BestBetta := 0;

      CopyRegion(SourImage, BaseRegion, Reg * RegionSize, Reg * RegionSize,
SourWidth);

      // Перебираємо домени
      for Dom := 0 to YDomains - 1 do
        for Dom := 0 to XDomains - 1 do
        begin

          // Визначаємо різницю в яскравості. Вона завжди одна для будь-яких
трансформацій.
          Betta := Regions[Reg, Reg].MeanColor - Domains[Dom, Dom].MeanColor;

          DCoord := Dom * DomainOffset;
          DCoord := Dom * DomainOffset;

          // Проходимо цикл по трансформаціях
          for TransNum := 0 to TransCount - 1 do
          begin
            // Виконуємо афінне перетворення
            TransformRegion(BaseRegion, Region, TTransformType(TransNum));

            // Визначаємо величину різниці між зображеннями
            Error := GetDifference(Region, DCoord, DCoord, Betta);

            // Запам'ятовуємо в тимчасові змінні кращі показники
            if Error < BestError then
            begin
```

```
              BestError := Error;
              BestFormNum := TransNum;
              BestDom := DCoord;
              BestDom := DCoord;
              BestBetta := Betta;
            end;

            if FStop then goto LExit; // Миттєва реакція на команду виходу Stop
          end;  // Цикл по трансформаціях
        end;  // Цикл по доменам

      // Тепер відомо все, що потрібно для даного регіону
      with Regions[Reg, Reg].Ifs do
      begin
        DomCoord := BestDom;
        DomCoord := BestDom;
        Betta := BestBetta;
        if BestFormNum = 1 then BestFormNum := 3 else // 90 -> 270
          if BestFormNum = 3 then BestFormNum := 1;  // 270 -> 90
        FormNum := BestFormNum;
      end;


      if Reg * XRegions + Reg = 10 then
      begin
        OneRegTime := (GetTickCount - Tc) div 10;
        AllRegTime := OneRegTime * Cardinal(XRegions * YRegions);
      end;
      if Assigned(BackProc) and (Percent >= 0) then
        BackProc(Trunc(Percent), (AllRegTime - OneRegTime * Cardinal(Reg *
XRegions + Reg)) div 1000);

    end; // Цикл по регіонах

  FIfsIsLoad := True;

  LExit:

  FreeMem(BaseRegion);
  FreeMem(Region);
end;

constructor TFractal.Create(AOwner: TComponent);
begin
  inherited;
  FRegionSize := 8;
  DomainOffset := 1;
  Gamma := 0.75;
  MaxImageSize := 512;
end;

procedure TFractal.CreateDomains;
var
  Y, X: Integer;
begin
  Domains := nil;

  SetLength(Domains, XDomains, YDomains);

  // Для кожного домену визначаємо його координати й усереднену яскравість
  for Y := 0 to YDomains - 1 do
    for X := 0 to XDomains - 1 do
      Domains[X, Y].MeanColor := GetMeanBrigth(DomainImage, X * DomainOffset,
        Y * DomainOffset, DomainImageWidth);
end;

procedure TFractal.CreateRegions;
var
  X, Y: Integer;
```

```
begin
  Regions := nil;
  SetLength(Regions, XRegions, YRegions);

  // Для кожного регіону визначаємо його координати й усереднену яскравість
  for Y := 0 to YRegions - 1 do
    for X := 0 to XRegions - 1 do
      Regions[X, Y].MeanColor := GetMeanBrigth(SourImage, X * RegionSize, Y *
RegionSize, SourWidth);
end;

destructor TFractal.Destroy;
begin
  ClearData();
  inherited;
end;

procedure TFractal.DrawImage(Image: TBitmap; Regions: Boolean = True);
var
  X, Y, Pixel: Integer;
  Handle: HDC;
begin
  if SourWidth * SourHeight < 1 then
    Error('Помилка відмальовування зображення !', []);
  Image.Width := SourWidth;
  Image.Height := SourHeight;
  Handle := Image.Canvas.Handle;

  for Y := 0 to SourHeight - 1 do
  begin
    for X := 0 to SourWidth - 1 do
    begin
      Pixel := SourImage[Y * SourWidth + X];
     Pixel := (Pixel shl 16) + (Pixel shl 8) + Pixel;
      SetPixel(Handle, X, Y, Pixel);
    end;
  end;

  if not Regions then
  for Y := 0 to SourHeight div 2 - 1 do
  begin
    for X := 0 to SourWidth div 2 - 1 do
    begin
      Pixel := DomainImage[Y * DomainImageWidth + X];
      Pixel := (Pixel shl 16) + (Pixel shl 8) + Pixel;
      SetPixel(Handle, X, Y, Pixel);
    end;
  end;
end;

procedure TFractal.Error(Msg: string; Args: array of const);
begin
  raise Exception.CreateFmt(Msg, Args);
end;

function TFractal.GetMeanBrigth(Image: PByteArray; X, Y, Width: Integer): Byte;
var
  I, J, Bufer: Integer;
begin
  Bufer := 0;
  for I := Y to Y + RegionSize - 1 do
    for J := X to X + RegionSize - 1 do
      Inc(Bufer, Image[I * Width + J]);
  Result := Trunc(Bufer / (RegionSize * RegionSize));
end;

procedure TFractal.LoadImage(Image: TBitmap);
var
  X, Y: Integer;
```

```
  PixColor: TColor;
  red, green, blue, mask: integer;
begin
  ClearData;  // Видаляємо масиви

  SourWidth := (Image.Width div RegionSize) * RegionSize;
  SourHeight := (Image.Height div RegionSize) * RegionSize;
  if (SourWidth > MaxImageSize) or (SourWidth < 16) or
     (SourHeight > MaxImageSize) or (SourHeight < 16)
    then Error('Неприпустимі розміри зображення %d x %d', [Image.Width,
Image.Height]);

  // ======= Заповнюємо масив SourImage (для регіонів) ===========
  // Виділяємо пам'ять під зображення
  GetMem(SourImage, SourWidth * SourHeight);

  // Робимо пікселі сірими й зберігаємо їх у строковому масиві SourImage
  mask := $000000FF;
  for Y := 0 to SourHeight - 1 do
    for X := 0 to SourWidth - 1 do
    begin
      PixColor := Image.Canvas.Pixels[X, Y]; // Визначаємо колір пікселя
      red := (PixColor shr 16)and mask;
      green := (PixColor shr 8)and mask;
      blue := PixColor and mask;
      SourImage[Y * SourWidth + X] := Byte((red + green + blue) div 3);
  //SourImage[Y * SourWidth + X] :=PixColor;
    end;
  // Всі! Тепер всі пікселі стали сірими.

  // ======= Заповнюємо масив DomenImage (для доменів) ===========
  // Взагалі-Те домени в 2 рази більше регіонів, однак через цього їх складно
порівнювати.
  // А от якщо ми доменне зображення зменшимо в 4 рази (по площі), то
  // розмір 1 домену стане рівним розміру 1 регіону, що набагато краще
  // і ощадливіше.
  CreateDomainImage;

  FIfsIsLoad := False;
end;

procedure TFractal.SetDomainOffset(const Value: Integer);
begin
  if (Value < 1) or (Value > 32) then
    Error('Задана неприпустима величина зсуву домену %d', [Value]);
  FDomainOffset := Value;
end;

procedure TFractal.SetGamma(const Value: Real);
begin
  if (Value < 0.1) or (Value > 1) then
    Error('Параметр GAMMA має неприпустиме значення %d', [Value]);
  FGamma := Value;
end;

procedure TFractal.SetMaxImageSize(const Value: Integer);
begin
  FMaxImageSize := Value;
end;

procedure TFractal.SetRegionSize(const Value: Integer);
begin
  if (Value < 2) or (Value > 64) then
    Error('Задане неприпустиме значення регіону %d', [Value]);
  FRegionSize := Value;
end;

function TFractal.XDomains: Integer;
begin
```

```
  Result := SourWidth div (2 * DomainOffset) - 1;
  if Result <= 1 then
    Error('Неприпустима кількість доменів по X %d', [Result]);
end;

function TFractal.YDomains: Integer;
begin
  Result := SourHeight div (2 * DomainOffset) - 1;
  if Result <= 1 then
    Error('Неприпустима кількість доменів по Y %d', [Result]);
end;

function TFractal.XRegions: Integer;
begin
  Result := SourWidth div RegionSize;
end;

function TFractal.YRegions: Integer;
begin
  Result := SourHeight div RegionSize;
end;

procedure TFractal.TransformRegion(Sour, Dest: PByteArray; TransformType:
TTransformType);
var
  I, J: Integer;
begin
  case TransformType of
    ttRot0: // Поворот на 0 градусів
      begin
        for I := 0 to RegionSize - 1 do
          for J := 0 to RegionSize - 1 do
            Dest[I * RegionSize + J] := Sour[I * RegionSize + J];
      end;

    ttRot90: // Поворот на 90 градусів
      begin
        for I := 0 to RegionSize - 1 do
          for J := 0 to RegionSize - 1 do
            Dest[(RegionSize - 1 - J) * RegionSize + I] := Sour[I * RegionSize +
J];
      end;

    ttRot180: // Поворот на 180 градусів
      begin
        for I := 0 to RegionSize - 1 do
          for J := 0 to RegionSize - 1 do
            Dest[(RegionSize - 1 - I) * RegionSize + (RegionSize - 1 - J)] :=
Sour[I * RegionSize + J];
      end;

    ttRot270: // Поворот на 270 градусів
      begin
        for I := 0 to RegionSize - 1 do
          for J := 0 to RegionSize - 1 do
            Dest[J * RegionSize + (RegionSize - 1 - I)] := Sour[I * RegionSize +
J];
      end;

    ttSimm: // Симетрія відносно X
      begin
        for I := 0 to RegionSize - 1 do
          for J := 0 to RegionSize - 1 do
            Dest[(RegionSize - 1 - I) * RegionSize + J] := Sour[I * RegionSize +
J];
      end;

    ttSimm: // Симетрія відносно B
      begin
```

```
        for I := 0 to RegionSize - 1 do
          for J := 0 to RegionSize - 1 do
            Dest[I * RegionSize + (RegionSize - 1 - J)] := Sour[I * RegionSize +
J];
      end;

    ttSimmDiag1: // Симетрія від. головної діагоналі
      begin
        for I := 0 to RegionSize - 1 do
          for J := 0 to RegionSize - 1 do
            Dest[J * RegionSize + I] := Sour[I * RegionSize + J];
      end;

    ttSimmDiag2: // Симетрія від. другорядної діагоналі
      begin
        for I := 0 to RegionSize - 1 do
          for J := 0 to RegionSize - 1 do
            Dest[(RegionSize - 1 - J) * RegionSize + (RegionSize - 1 - I)] :=
Sour[I * RegionSize + J];
      end;
  end;
end;

function TFractal.DomainImageWidth: Integer;
begin
  Result := SourWidth div 2;
end;

procedure TFractal.LoadFromFile(FileName: string);
var
  X, Y: Integer;
  Header: TIfsHeader;
begin
  if not FileExists(FileName) then
    Error('Файл "%s" не існує', [FileName]);

  with TMemoryStream.Create do
  begin
    LoadFromFile(FileName);
    Seek(0, soFromBeginning);
    Read(Header, SizeOf(TIfsHeader));
    if Header.FileExt <> 'IFS' then
    begin
      Free;
      Error('Файл "%s" має неприпустимий формат!', [FileName]);
    end;

    SourWidth := Header.XRegions * Header.RegSize;
    SourHeight := Header.YRegions * Header.RegSize;
    RegionSize := Header.RegSize;

    Regions := nil;

    SetLength(Regions, XRegions, YRegions);
    for Y := 0 to YRegions - 1 do
      for X := 0 to XRegions - 1 do
        Read(Regions[X, Y].Ifs, SizeOf(TIfsRec));

    Free;
  end;

  // Потрібний для масштабування при декомпресії
  FBaseRegionSize := RegionSize;

  FIfsIsLoad := True;
end;

procedure TFractal.SaveToFile(FileName: string);
var
```

```
  X, Y: Integer;
  Header: TIfsHeader;
begin
  if Regions = nil then
    Error('Стиск зображення не виконано!', []);

  if FileExists(FileName) and not DeleteFile(FileName) then
    Error('Неможливо видалити файл %s. Можливо він використовується іншим
додатком' +
          'або доступний тільки для читання.', [FileName]);

  Header.FileExt := 'IFS';
  Header.RegSize := RegionSize;
  Header.XRegions := XRegions;
  Header.YRegions := YRegions;

  with TMemoryStream.Create() do
  begin
    // Зберігаємо заголовну інформацію
    Write(Header, SizeOf(TIfsHeader));
    for Y := 0 to YRegions - 1 do
      for X := 0 to XRegions - 1 do
        Write(Regions[X, Y].Ifs, SizeOf(TIfsRec));

    try
      SaveToFile(FileName);
    except
      Free;
      Error('Відбулася помилка при збереженні у файл "%s"', [FileName]);
    end;
    Free;
  end;

end;

procedure TFractal.Decompress(IterCount: Integer = 15; RegSize: Integer = 0);
var
  I, J, X, Y, Pixel, Iter: Integer;
  Domain1, Domain2: PByteArray;
  Scale: Real;
begin
  // Масив Region повинен бути вже заповненим.
  if not FIfsIsLoad then
    Error('Дані, необхідні для декомпресії, не завантажені!', []);


  Scale := 1;
  if RegSize >= 2 then
    begin
      SourWidth := XRegions * RegSize;
      SourHeight := YRegions * RegSize;
      Scale := FBaseRegionSize / RegSize;
      RegionSize := RegSize;
    end;


  // Створюємо сіре зображення.
  if Assigned(SourImage) then FreeMem(SourImage);
  GetMem(SourImage, SourWidth * SourHeight);

  // Робимо пікселі сірими й зберігаємо їх у строковому масиві SourImage
  for I := 0 to SourHeight * SourWidth - 1 do SourImage[I] := 127;

  for Iter := 1 to IterCount do
  begin
    // Створюємо доменне зображення
    CreateDomainImage;
    // Доменне й регіонне зображення створили
```

```
    // Проходимо по всіх регіонах
    for J := 0 to YRegions - 1 do
      for I := 0 to XRegions - 1 do
      begin
        // Запам'ятовуємо відповідних домен, щоб над ним можна було виконати
перетворення
        GetMem(Domain1, RegionSize * RegionSize);
        GetMem(Domain2, RegionSize * RegionSize);
        CopyRegion(DomainImage, Domain1,
          Trunc(Regions[I, J].Ifs.DomCoord / Scale),
          Trunc(Regions[I, J].Ifs.DomCoord / Scale), DomainImageWidth);

        // Виконуємо задане перетворення
        TransformRegion(Domain1, Domain2, TTransformType(Regions[I,
J].Ifs.FormNum));

        // Змінюємо пікселі поточного регіону
        for Y := 0 to RegionSize - 1 do
          for X := 0 to RegionSize - 1 do
          begin
            Pixel := Domain2[Y * RegionSize + X] + Regions[I, J].Ifs.Betta;
            SourImage[(J * RegionSize + Y) * SourWidth + I * RegionSize + X] :=
Pixel;
          end;

        FreeMem(Domain1);
        FreeMem(Domain2);
      end;
  end;
end;

procedure TFractal.CreateDomainImage;
var
  X, Y, PixColor: Integer;
begin
  if Assigned(DomainImage) then FreeMem(DomainImage);
  GetMem(DomainImage, SourWidth * SourHeight div 4);

  for Y := 0 to SourHeight div 2 - 1 do
    for X := 0 to SourWidth div 2 - 1 do
    begin
      // Визначаємо усереднений колір пікселя (по кольорам 4-х сусідніх
пікселів)
      PixColor :=
        SourImage[Y * 2 * SourWidth + X * 2] + SourImage[Y * 2 * SourWidth + X *
2 + 1] +
        SourImage[(Y * 2 + 1) * SourWidth + X * 2] + SourImage[(Y * 2 + 1) *
SourWidth + X * 2 + 1];
      DomainImage[Y * DomainImageWidth + X] := Trunc(PixColor / 4 * Gamma);
    end;
end;

function TFractal.GetDifference(Region: PByteArray; DomCoord,
  DomCoord, Betta: Integer): Integer;
var
  X, Y, Diff: Integer;
begin
  Result := 0;
  for Y := 0 to RegionSize - 1 do
    for X := 0 to RegionSize - 1 do
    begin
      Diff := Region[Y * RegionSize + X] -
        DomainImage[(DomCoord + Y) * DomainImageWidth + DomCoord + X];

      Inc(Result, Sqr(Abs(Diff - Betta)));
    end;
end;

procedure TFractal.CopyRegion(AllImage, Dest: PByteArray; X, Y,
```

```
  Width: Integer);
var
  I, J: Integer;
begin
  for I := 0 to RegionSize - 1 do
    for J := 0 to RegionSize - 1 do
      Dest[I * RegionSize + J] := AllImage[(Y + I) * Width + X + J];
end;

procedure TFractal.BuildImageWithDomains;
var
  I, J, X, Y: Integer;
  Domain1, Domain2: PByteArray;
begin
  if not FIfsIsLoad then
    Error('Дані, необхідні для відновлення по доменам, не завантажені!', []);

  for J := 0 to YRegions - 1 do
    for I := 0 to XRegions - 1 do
    begin
      GetMem(Domain1, RegionSize * RegionSize);
      GetMem(Domain2, RegionSize * RegionSize);

      // Копіюємо домен
      CopyRegion(DomainImage, Domain1, Regions[I, J].Ifs.DomCoord,
        Regions[I, J].Ifs.DomCoord, DomainImageWidth);

      // Виконуємо афінне перетворення
      TransformRegion(Domain1, Domain2, TTransformType(Regions[I,
J].Ifs.FormNum));

      // Копіюємо домен у регіон
      for Y := 0 to RegionSize - 1 do
        for X := 0 to RegionSize - 1 do
          SourImage[(J * RegionSize + Y) * SourWidth + I * RegionSize + X] :=
            Domain2[Y * RegionSize + X] + Regions[I, J].Ifs.Betta;

      FreeMem(Domain1);
      FreeMem(Domain2);
    end;
end;

procedure TFractal.Stop;
begin
  FStop := True;
end;

function TFractal.GetPixel(X, Y: Integer): Byte;
begin
  Result := SourImage[Y * SourWidth + X];
end;

function TFractal.GetIFSFileSize: Cardinal;
begin
  Result := (ImageWidth div RegionSize) * (ImageHeight div RegionSize) *
SizeOf(TIfsRec);
  if Result > 0 then Inc(Result, SizeOf(TIfsHeader));
end;

end.
```

FractComp.dpr - файл проекту

```
program FractComp;

uses
  Forms,
  U_Main in 'U_Main.pas' {Form1},
  about in 'about.pas' {Form2},
  U_ShowImage in 'U_ShowImage.pas' {fmShowImage},
  FractalCompression in 'FractalCompression.pas';

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.Run;
end.
```

U_Main.pas - файл головної програми

```
unit U_Main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
about,
  Dialogs, StdCtrls, ExtCtrls, jpeg, ExtDlgs, FractalCompression, ComCtrls,
  Menus;

type
  TForm1 = class(TForm)
    ImPreview: TImage;
    Edit1: TEdit;
    Button1: TButton;
    Button2: TButton;
    OpenPictureDialog1: TOpenPictureDialog;
    lbSize: TLabel;
    CheckBox1: TCheckBox;
    ProgressBar1: TProgressBar;
    Button4: TButton;
    Button5: TButton;
    Label4: TLabel;
    Edit2: TEdit;
    Label5: TLabel;
    Edit3: TEdit;
    Button3: TButton;
    OpenDialog1: TOpenDialog;
    Button6: TButton;
    Label6: TLabel;
    Edit4: TEdit;
    Button7: TButton;
    Button8: TButton;
    CheckBox2: TCheckBox;
    Label8: TLabel;
    Edit5: TEdit;
    Label9: TLabel;
    lbTime: TLabel;
    Button9: TButton;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    Label2: TLabel;
    Label7: TLabel;
    Label3: TLabel;
    Image1: TImage;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Button10: TButton;
    Label10: TLabel;
    Label11: TLabel;
    Edit6: TEdit;
    Button11: TButton;
    SaveDialog1: TSaveDialog;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
```

```
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure Button10Click(Sender: TObject);
    procedure ImPreviewClick(Sender: TObject);
    procedure Image1Click(Sender: TObject);
    procedure Button11Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    ImageExists: Boolean;
    procedure ProgressProc(Percent: Integer; TimeRemain: Cardinal);
  end;

var
  Form1: TForm1;
  FractalComp: TFractal;
  FractalDeComp: TFractal;

implementation

uses U_ShowImage;

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var SearchRec: TSearchRec;  t:extended;  tt:integer;
f: file of Byte;     size: Longint;


begin
  if OpenPictureDialog1.Execute then
  begin

    try
      ImPreview.Picture.LoadFromFile(OpenPictureDialog1.FileName);
      if (ImPreview.Picture.Width > 512) or (ImPreview.Picture.Height > 512)
then Abort;
      lbSize.Caption := Format('%d x %d', [ImPreview.Picture.Width,
ImPreview.Picture.Height]);
       label11.Caption := lbSize.Caption;
Кбайт');

      FindFirst(OpenPictureDialog1.FileName, faAnyFile, SearchRec);
      t:=((SearchRec.Size*100 )div (1024))/100;

 Label17.Caption := floattostr(t) + ' Кбайт';
 FindClose(SearchRec);



      ImageExists := True;

      // Завантажуємо зображення в об'єкт винятково з метою реалізації
      // операції передперегляду
      FractalComp.LoadImage(ImPreview.Picture.Bitmap);
      Edit1.Text := OpenPictureDialog1.FileName;
    except
      ImPreview.Picture := nil;
      ImPreview.Canvas.TextOut(5, 10, 'Помилка');
      ImPreview.Canvas.TextOut(5, 30, 'завантаження');
      lbSize.Caption := '';
      ImageExists := False;


    end;
  end;
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ImPreview.Canvas.TextOut(5, 10, '');
  ImPreview.Canvas.TextOut(5, 30, '');

  Image1.Canvas.TextOut(5, 10, '');
  Image1.Canvas.TextOut(5, 30, '');

  FractalComp := TFractal.Create(Application);
  FractalDeComp := TFractal.Create(Application);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  // Показує зображення на екрані повністю.
  if ImageExists then
  begin
    Application.CreateForm(TfmShowImage, fmShowImage);
    if CheckBox1.Checked then
      FractalComp.DrawImage(fmShowImage.Image1.Picture.Bitmap, False)
    else
      fmShowImage.Image1.Picture := ImPreview.Picture;

    with fmShowImage do
    begin
      AutoSize := True;
      Position := poScreenCenter;
      ShowModal;
    end;

    FreeAndNil(fmShowImage);
  end;
end;

procedure TForm1.ProgressProc(Percent: Integer; TimeRemain: Cardinal);
begin
  ProgressBar1.Position := Percent;
  lbTime.Caption := IntToStr(TimeRemain);
  Application.ProcessMessages;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  if ImageExists then
  begin
    FractalComp.RegionSize := StrToInt(Edit3.Text);
    FractalComp.DomainOffset := StrToInt(Edit2.Text);

    FractalComp.LoadImage(ImPreview.Picture.Bitmap);
    FractalComp.Compress(CheckBox2.Checked, ProgressProc);

    ProgressBar1.Position := 0;
    lbTime.Caption := '';
     FractalComp.BuildImageWithDomains;


  FractalComp.DrawImage(Image1.Picture.Bitmap);
  label15.Caption:=(floatToStr(((FractalComp.GetIFSFileSize*100 )div 1024)/100
) + ' Кбайт');
  end;
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
  OpenDialog1.Title := 'Збереження IFS-даних';
  if OpenDialog1.Execute then
    FractalComp.SaveToFile(OpenDialog1.FileName);
end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  OpenDialog1.Title := 'Завантаження IFS-даних';
  if OpenDialog1.Execute then
    FractalDeComp.LoadFromFile(OpenDialog1.FileName);
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
   OpenDialog1.Title := 'Завантаження IFS-даних';
  if OpenDialog1.Execute then   begin
    FractalDeComp.LoadFromFile(OpenDialog1.FileName);

  FractalDeComp.Decompress(StrToInt(Edit4.Text), StrToInt(Edit5.Text));
  // Показує зображення на екрані повністю.

Application.CreateForm(TfmShowImage, fmShowImage);
  FractalDeComp.DrawImage(Image2.Picture.Bitmap);
  end;

FractalComp.DrawImage(Image2.Picture.Bitmap);

  with fmShowImage do
  begin
    AutoSize := True;
    Position := poScreenCenter;
ShowModal;
end;

  FreeAndNil(fmShowImage);

end;

procedure TForm1.Button7Click(Sender: TObject);
begin
  FractalComp.BuildImageWithDomains;

  FractalComp.DrawImage(Image1.Picture.Bitmap);


end;

procedure TForm1.Button8Click(Sender: TObject);
begin
  FractalComp.Stop;
end;

procedure TForm1.Button9Click(Sender: TObject);
begin
  ShowMessage(floatToStr(((FractalComp.GetIFSFileSize*100 )div 1024)/100  ) + '
байт');
end;

procedure TForm1.Button10Click(Sender: TObject);
begin
Image1.Picture.SaveTofile('1.bmp');

end;

procedure TForm1.ImPreviewClick(Sender: TObject);
begin
 // Показує зображення на екрані повністю.
  if ImageExists then
  begin
    Application.CreateForm(TfmShowImage, fmShowImage);
    if CheckBox1.Checked then
      FractalComp.DrawImage(fmShowImage.Image1.Picture.Bitmap, False)
    else
```

```
      fmShowImage.Image1.Picture := ImPreview.Picture;

    with fmShowImage do
    begin
      AutoSize := True;
      Position := poScreenCenter;
      ShowModal;
    end;

    FreeAndNil(fmShowImage);
  end;
end;

procedure TForm1.Image1Click(Sender: TObject);
begin
// Показує зображення на екрані повністю.
  if ImageExists then
  begin
    Application.CreateForm(TfmShowImage, fmShowImage);
    if CheckBox1.Checked then
      FractalComp.DrawImage(fmShowImage.Image1.Picture.Bitmap, False)
    else
      fmShowImage.Image1.Picture := Image1.Picture;

    with fmShowImage do
    begin
      AutoSize := True;
      Position := poScreenCenter;
      ShowModal;
    end;

    FreeAndNil(fmShowImage);
  end;
end;

procedure TForm1.Button11Click(Sender: TObject);
begin
 if saveDialog1.Execute  then   begin
 Edit6.Text :=saveDialog1.FileName;
 FractalComp.SaveToFile(saveDialog1.FileName);
 end;
end;

end.
```

about.pas - файл довідки

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls;

type
  TForm2 = class(TForm)
    Image1: TImage;
    BitBtn1: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.BitBtn1Click(Sender: TObject);
begin
Form2.Close;
end;

end.
```