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 стиску зображень за допомогою фракталів”

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

Керівник проекту
кандидат технічних наук, доцент
_____ Доренський О.П.
«____» _____ 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

Мартинову Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи стиску зображень за допомогою фракталів*

2. Керівник роботи *Доренський Олександр Павлович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи стиску зображень за допомогою фракталів*

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

Мартинов Д.О. Дослідження та програмна реалізація системи стиску зображень за допомогою фракталів. 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 стиску зображень за допомогою фракталів.

Метою розробки є дослідження та програмна реалізація системи стиску зображень за допомогою фракталів.

Об'єктом дослідження є процес стиску зображень за допомогою фракталів.

Предметом дослідження є методи стиску зображень за допомогою фракталів.

Методи дослідження базуються на методах обробки зображень, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи стиску зображень за допомогою фракталів.

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 RAD Studio Delphi 10.4.

Ключові слова: комп'ютерні науки, стиск зображень, фракталів

ABSTRACT

Martynov D.O. Research and software implementation of image compression system using fractals. 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 image compression system using fractals.

The goal of development is research and software implementation of image compression system using fractals.

The object of research is the process of image compression using fractals.

The subject of research is image compression methods using fractals.

Research methods are based on image processing methods, mathematical statistics methods, and software development methods.

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

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 RAD Studio Delphi 10.4 environment.

Keywords: computer science, image compression, fractals

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

ВСТУП

Актуальність теми. Останнім часом зображення й ілюстрації стали використовуватися повсюдно. Проблема, пов'язана з великим обсягом для їхньої обробки й зберігання, з'явилася при роботі й на робочих станціях, і на персональних комп'ютерах. Розроблено велику кількість різних алгоритмів архівації графіки.

Майкл Барнслі й Алан Слоун знайшли новий метод рішення даного завдання. У методі використовується принципово нова ідея – не близькість кольорів у локальній області, а подоба різних по розміру областей зображення. Так за допомогою стандартних прийомів обробки зображень, таких, як виділення країв і аналіз текстурних варіацій, зображення ділиться на сегменти й кодується за допомогою деякого стискаючого афінного перетворення. Відновлення зображення відбувається за допомогою багаторазового застосування цього афінного перетворення.

Метою цієї роботи є побудова методу фрактального стиску для статичних зображень із використанням генетичних алгоритмів (ГА). У роботі розглядаються основні принципи методу, його обґрунтування, кілька алгоритмів його реалізації й модифікація генетичного алгоритму в застосуванні до завдання пошуку перетворення, що кодує.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи стиску зображень за допомогою фракталів.

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

– Огляд існуючих систем стиску зображень за допомогою фракталів.

– Дослідження системи стиску зображень за допомогою фракталів.

– Програмна реалізація системи стиску зображень за допомогою фракталів.

Об'єктом дослідження є процес стиску зображень за допомогою фракталів.

Предметом дослідження є методи стиску зображень за допомогою фракталів.

Методи дослідження базуються на методах обробки зображень, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод стиску зображень за допомогою фракталів.

– Розроблено вітчизняний продукт стиску зображень за допомогою фракталів, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі стиску зображень за допомогою фракталів.

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 стиску зображень за допомогою фракталів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для стиску зображень за допомогою фракталів. Існують наступні визначення фракталів:

– Фракта́л – складна геометрична фігура, що володіє властивістю самоподоби, тобто складена з декількох частин, кожна з яких подібна до всієї фігури цілком. У більше широкому змісті під фракталами розуміють безлічі точок в евклідовому просторі, що мають дробову метричну розмірність (у змісті Мінковського або Хаусдорфа), або метричну розмірність, відмінну від топологічної.

– Фрактал – це нескінченно самоподібна геометрична фігура, кожний фрагмент якої повторюється при зменшенні масштабу.

– Фрактал – самоподібна безліч нецілої розмірності.

Існує наступне застосування фракталів в інформаційних технологіях:

– Стиск зображень. Існують алгоритми стиску зображення за допомогою фракталів. Вони засновані на ідеї про те, що замість самого зображення можна зберігати стискаюче відображення, для якого це зображення (або деяке близьке до нього) є нерухливою точкою. Один з варіантів даного алгоритму був використаний фірмою Microsoft при виданні своєї енциклопедії, але великого поширення ці алгоритми не одержали.

– Комп'ютерна графіка. Фрактали широко застосовуються в комп'ютерній графіці для побудови зображень природних об'єктів, таких, як дерева, кущі, гірські ландшафти, поверхні морів і так далі. Існує безліч програм, що служать для генерації фрактальних зображень.

– Децентралізовані мережі. Система призначення IP-адрес у мережі Netsukuku використовує принцип фрактального стиску інформації для компактного збереження інформації про вузли мережі. Кожний вузол мережі Netsukuku зберігає

всього 4 Кб інформації про стан сусідніх вузлів, при цьому будь-який новий вузол підключається до загальної мережі без необхідності в центральному регулюванні роздачі IP-адрес, що, наприклад, характерно для мережі Інтернет. Таким чином, принцип фрактального стиску інформації гарантує повністю децентралізовану, а отже, максимально усталену роботу всієї мережі.

Алгоритм фрактального стиску

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. Уперше можливість застосування теорії систем ітеруємих функцій до проблеми стиску зображення була досліджена Майклом Барнслі й Аланом Слоуном. Вони запатентували свою ідею в 1990 і 1991 р.р. (U.S. Patent 5 065 447). А. Жакен представив метод фрактального кодування, у якому використовуються системи доменних і рангових блоків зображення, блоків квадратної форми, що покривають все зображення. Цей підхід став основою для більшості методів фрактального кодування, застосовуваних сьогодні. Він був удосконалений Ювалом Фішером і рядом інших дослідників.

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.

На даний момент відомо досить велика кількість алгоритмів оптимізації перебору, що виникає при фрактальному стиску, оскільки більшість статей, що досліджували алгоритм були присвячені цій проблемі, і під час активних досліджень (1992 – 1996 року) виходило до 300 статей у рік. Найбільш ефективними виявилися два напрямки досліджень: метод виділення особливостей (feature extraction) і метод класифікації доменів (classification of domains).

1.2 Область застосування

Областю застосування розробляємого, у результаті виконання магістерського проектування, програмного забезпечення є стиск зображень. Стиск зображень – застосування алгоритмів стиску даних до зображень, що зберігається в цифровому виді. У результаті стиску зменшується розмір зображення, через що зменшується час передачі зображення по мережі й заощаджується простір для зберігання. Стиск зображень підрозділяють на стиск із втратами якості й стиск без втрат. Стиск без втрат часто більш сприятливий для штучно побудованих зображень, таких як графіки, іконки програм, або для спеціальних випадків, наприклад, якщо зображення призначені для наступної обробки алгоритмами розпізнавання зображень. Алгоритми стиску із втратами при збільшенні ступеня стиску як правило породжують добре помітні людському оку артефакти.

У рамках комп'ютерної графіки бурхливо розвивається зовсім нова область – алгоритми архівації 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 архівації 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 архівації 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.

Усього на даний момент відомо мінімум три сімейства алгоритмів, які розроблені винятково для стиску зображень, і застосовувані в них методи практично неможливо застосувати до архівації ще яких-небудь видів даних.

Для того, щоб говорити про алгоритми стиску зображень, ми повинні визначитися з декількома важливими питаннями:

1. Які критерії ми можемо запропонувати для порівняння різних алгоритмів?

2. Які класи зображень існують?

3. Які класи додатків, що використовують алгоритми компресії графіки, існують, і які вимоги вони пред'являють до алгоритмів?

Розглянемо ці питання докладніше.

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 архівації 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: відсканові 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 архівації й розархівації. 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 архівації 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 архівації й часу розархівації. Час архівації 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 архівації. 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 розархівації. 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.

У геоінформаційних системах – при зберіганні аерофотознімків місцевості – специфічними проблемами є великий розмір зображення й необхідність вибірки лише частини зображення на вимогу. Крім того, може знадобитися масштабування. 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 архівації. 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.

Вимоги додатків до алгоритмів компресії

У попередньому розділі ми визначили, які додатки є споживачами алгоритмів архівації зображень. Однак помітимо, що додаток визначає характер використання зображень (або велика кількість зображень зберігається й використовується, або зображення заванатжуються по мережі, або зображення великі по розмірах, і нам необхідна можливість одержання лише частини...). 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 архівації, 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 архівації 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 архівації? 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.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи стиску зображень за допомогою фракталів, 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

Розглянемо програми, які дозволяють здійснювати стиск зображень.

FastStone Image Viewer

FastStone Image Viewer – чудовий приклад безкоштовного графічного переглядача й редактора із простим інтерфейсом. Запустивши FastStone Image Viewer, відкриваємо папку з нашими фотографіями й виділяємо лівою кнопкою миші потрібне зображення. Можна вибрати відразу декілька фото, утримуючи клавішу ALT. Далі вибираємо команду Файл – Email або тиснемо значок на панелі інструментів. Відкривається вікно стиску фото.

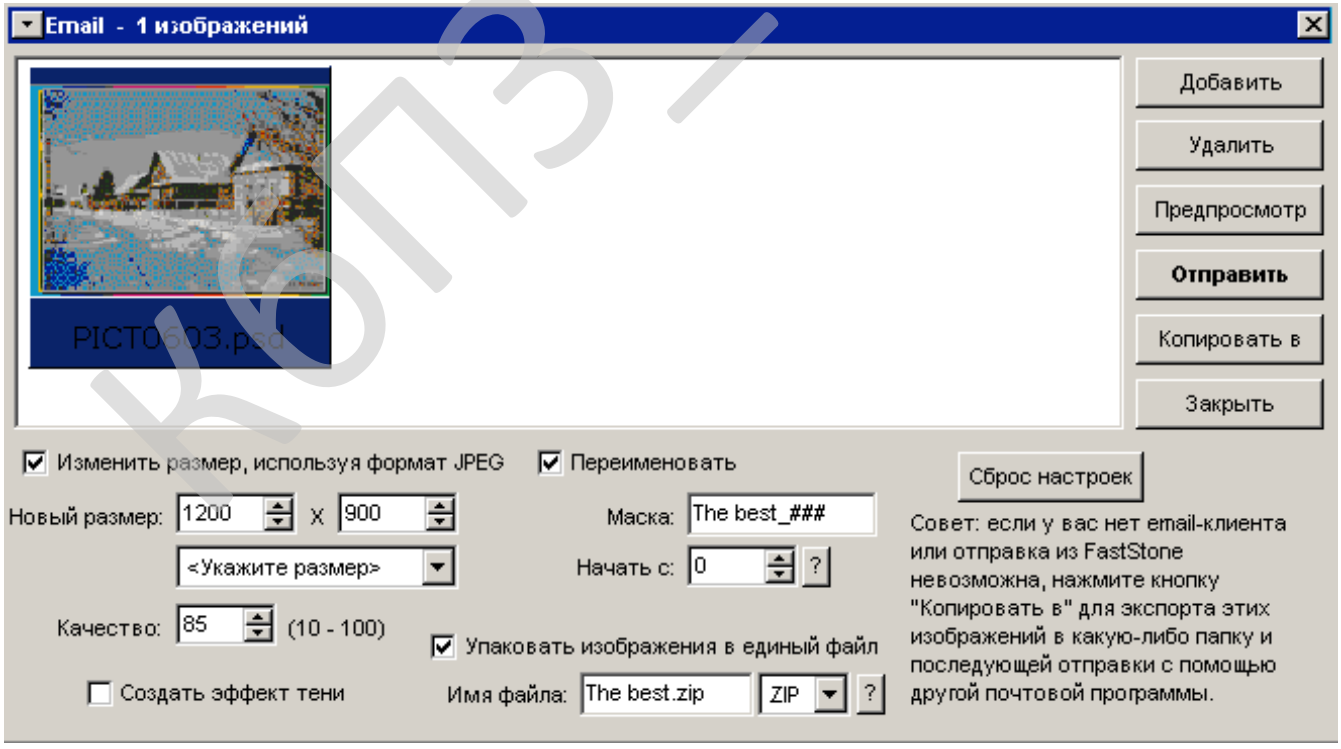

Рисунок 2.1 – Інтерфейс користувача FastStone Image Viewer

Розглянемо пропоновані параметри оптимізації зображення:

– Новий розмір – зміна розміру фотографії. Для перегляду на екрані монітора в гарній якості цілком достатньо від 640x480 до 1200x900 пікселів. Потрібне значення вибираєте зі списку, що випадає, або встановлюєте вручну. Чим більше розмір, тим більше буде обсяг файлу стислого фото. Необхідне для Інтернету розрішення 72 пікс/дюйм програма встановить автоматично.

– Якість – від вибору цього параметра також залежить обсяг отриманого файлу – чим менше значення, тим краще. Для контролю якості стислого зображення переглянете картинку, розгорнувши її на повний екран – кнопка "Передперегляд".

– Створити ефект тіні – прикраса: фотографія міститься на білому тлі з тінню, що створює об'ємний ефект.

– Перейменувати – створеним файлам можна задати нове ім'я, а програма буде їх послідовно нумерувати.

– Упакувати зображення в єдиний файл – всі фотографії після оптимізації записуються в один файл, що зручніше додати до листа, ніж кілька окремих файлів. Передбачено впакування в ZIP або EXE формати. Якщо ви не впевнені, що в одержувача електронної пошти є програма-архіватор, то збережете фото в EXE-форматі, що саморозпаковується.

Після завдання всіх параметрів можна відразу відіслати фото по електронній пошті (кнопка "Відправити") або зберегти оптимізовані зображення в окремій папці. Другий варіант переважніший, тому що ви завжди будете мати під рукою готові для відправлення файли, що займають небагато місця на диску. Для збереження оптимізованих зображень тиснемо кнопку "Копіювати в". Програма підготує зображення й попросить указати папку для збереження результатів. Якщо такої ще немає, то можете її тут же й створити. Потім знову тиснемо "Копіювати", і файл із оптимізованим зображенням або фотографіями готовий для пересилання друзям і знайомим, а також для розміщення в Інтернеті.

RIOT

RIOT, у перекладі з англійського означає Інструмент Радикальної Оптимізації Зображень, безкоштовний програмний продукт, що дозволяє докладно настроїти стиск і при цьому візуально редагувати настроювання. Програма використовує два вікна, в одному з яких відображається зображення оригіналу а в другому вужі стисле при певних настроюваннях, що властиво й відрізняє цю програму від більшості подібних. Бувають програми в яких стиск відбувається із певними параметрами зображення а підсумок цієї дії не видно. А тут можна подивитися й у реальному часі подивитися на зміни, що дозволяє домогтися максимального ефекту від стиску будь-якого зображення.

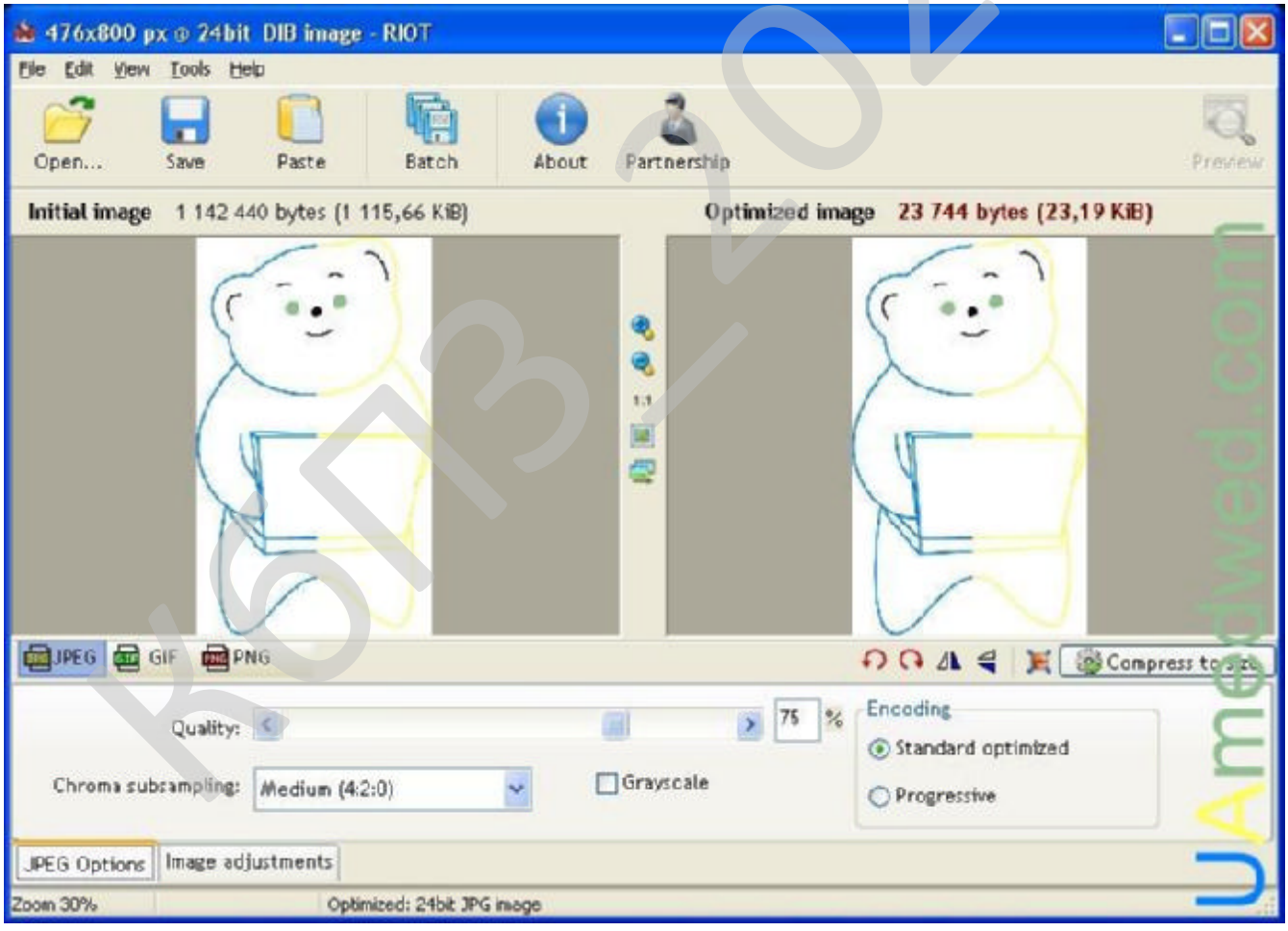

Рисунок 2.2 – Інтерфейс користувача RIOT

Ця програма не тільки якісно стискає але вона ще й швидко працює, що властиво немаловажно при роботі в мережі Інтернет. Як плюс можна відзначити той факт що програма може перетворювати ваше вихідне зображення в JPG, GIF, PNG.

RIOT доступно як окремий додаток так і як плагін до таких програмних продуктів як: GIMP, IrfanView і XnView.

Основні характеристики:

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

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

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

– можна виставити певний розмір файлу, для стисливого зображення, на екрані відразу відображається розмір підсумкового файлу й вихідного;

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

– є можливість роботи із прозорістю;

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

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

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

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

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

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

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

– розроблювачі затверджують що результати стиску цілком можна зрівняти з комерційними продуктами, і навіть набагато краще їх.

Деталі оптимізації

Для файлів JPEG доступні наступні настроювання:

– якість стиску;

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

– можна зберегти зображення як відтінки сірого 8-біт;

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

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

Настроювання для файлів GIF:

– зміна кольору;

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

– палітра 8 біт відтінки сірого;

– floyd-steinberg згладжування 1 біт монохромний;

– збереження черезрядкового.

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

Настроювання для файлів PNG:

– підтримуються вихідні кольори (24 біт RGB, 32 біт RGBA);

– зміна кольору:

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

– палітра 8 біт відтінки сірого;

– floyd-steinberg згладжування 1 біт монохромний;

– збереження черезрядкового;

– домогтися максимальних настроювань стиску можна шляхом використання інтеграції з популярними оптимізаторами PNG (optiPNG, PNGOut);

– додати / видалити / змінити зовнішній інструмент оптимізації PNG.

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

Настроювання для метаданих

Можна зберегти або видалити наступні метадані:

– Коментарі.

– Adobe XMP інформація.

– IPTC інформація.

– Профіль EXIF (у тому числі GPS і творець замітки).

– Профілі кольору ICC.

Невідомі або не підтримувані метадані автоматично видаляються.

Настроювання маски:

У програмі RIOT можна вибрати кілька варіантів прозорості:

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

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

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

Програма у своїй основі використовує вільно розповсюджувану бібліотеку для роботи із зображеннями FreeImage.

IrfanView

IrfanView – невеликий, але досить потужний додаток, призначений для перегляду й редагування графічних зображень. А додаткові модулі, які можна скачати із сайту розроблювача, значно розширюють список доступних функцій.

По-перше, розглянемо можливості IrfanView як редактору зображень. Ці можливості не безмежні й значно уступають надпотужному Adobe Photoshop, однак безліч корисних операцій із графікою можна виконати безпосередньо в IrfanView, не переходячи в зовнішній редактор.

Відкривши файл у вікні IrfanView, користувач одержує можливість:

– Перетворити зображення в монохромне (одноколірне) або негативне.

– Зробити колірну корекцію: змінити параметри яскравості, контрастності, насиченості, колірного балансу й т.д.

– Поміняти місцями кольори у використовуваній колірній моделі. Наприклад, RGB (Red, Green, Blue) можна перетворити в BGR (Blue, Green, Red). У цьому випадку пікселі червоного кольору стануть синіми, а сині – червоними. У результаті вийде досить ефектна картинка.

– Змінити різкість зображення, а також застосувати більше десяти додаткових візуальних ефектів, у тому числі забрати "червоні очі" з фотографії, перетворити її в рисунок маслом і т.п.

– Змінити розміри або повернути зображення на необхідне число градусів.

– Виділити деяку область зображення, вставити в неї текст, видалити й виконати деякі інші операції.

Такі основні можливості IrfanView як графічного редактора.

Друга сторона цієї програми – перегляд наявних зображень. Окрема функція відкриває вікно, у якому графічні файли зображені у вигляді мініатюр. З одного боку, нічого нового в цьому немає. Windows дозволяє бачити мініатюрні зображення й у своєму Провіднику, досить вибрати режим Ескізи сторінок. Але

погано те, що Windows створює мініатюри не занадто високої якості. Це виправдано, оскільки прискорює вивід безлічі мініатюр на екран. Однак в окремих випадках хотілося б бачити ці картинки в максимально доступній якості. От для цих випадків і потрібний IrfanView.

Наступна функція, що часто вимагається, – перегляд наявних зображень у режимі слайд-шоу. Тут все просто: вибираємо папку, у якій зберігаються зображення, указуємо, які типи файлів повинні брати участь у шоу, у якому порядку й з яким тимчасовим інтервалом їх потрібно виводити на екран, натискаємо кнопку Старт і любуємося результатом. Дуже зручно те, що будь-яке слайд-шоу можна зберегти у вигляді текстового файлу, щоб використовувати його надалі. Більше того, слайд-шоу може бути записане на компакт-диск у вигляді презентації або навіть збережено як екранна заставка. До речі, окрема функція програми дозволяє створювати із зображення шпалери для робочого стола.

IrfanView уміє виконувати пакетне перейменування й перетворення файлів з одного формату в інший. При перейменуванні до кожного імені файлу може автоматично додаватися будь-який текст, а також число-лічильник. Пакетне перетворення з одного формату в інший може супроводжуватися автоматичною зміною всіх параметрів зображення, на вибір користувача.

IrfanView підтримує роботу з TWAIN-пристроями, тому зображення може бути відскановано безпосередньо із цієї програми. Передбачено можливість створювати скріншот. При цьому користувач налаштовує наступні параметри:

– Робити знімок усього екрана, тільки активного вікна, або активного вікна без рядків заголовка й стану.

– Виконувати захват при натисканні гарячої клавіші (зазначеної користувачем) або періодично (через зазначений користувачем інтервал часу).

– Чи відображати курсор миші на знімку екрана.

– Показувати скріншот у вікні IrfanView або зберігати його на диску, у зазначеній користувачем папці, у тому або іншому форматі.

Крім усього перерахованого, IrfanView підтримує безліч додаткових модулів, що виконують всілякі завдання. От лише кілька прикладів таких плагінів:

– Effects містить набір додаткових візуальних ефектів для обробки зображень.

– Email дозволяє відправляти зображення по електронній пошті.

– IV-Player запускає програвач для прослуховування (перегляду) звукових і відео-файлів.

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 стиску зображень за допомогою фракталів.

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

Подання зображень кінцевим обсягом даних

Комп'ютерне зображення в його цифровому поданні є набором значень інтенсивностей світлового потоку, розподілених по кінцевій площі, що має звичайно прямокутну форму.

Для простоти розглянемо спочатку монохромні зображення. Тоді інтенсивність випромінюваної світлової енергії з одиниці поверхні в точці з координатами (ξ,η) зображення можна представити деяким числом $B(\xi,\eta)$. Одиничний елемент зображення, характеризуємий певним значенням (ξ,η), називається пікселім, а величина $z = f(\xi,\eta)$ – яскравістю.

Статистична й візуальна надмірність зображень

Потік даних про зображення має істотну кількість зайвої інформації, що може бути усунута практично без помітних для ока перекручувань.

Існує два типи надмірності.

– Статистична надмірність, пов'язана з кореляцією й передбачуваністю даних. Ця надмірність може бути усунута без втрати інформації, вихідні дані при цьому можуть бути повністю відновлені.

– Візуальна (суб'єктивна) надмірність, яку можна усунути із частковою втратою даних, що мало впливають на якість відтворених зображень; це – інформація, яку можна вилучити із зображення, не порушуючи візуально сприйману якість зображень.

Статистична надмірність зображень

Нехай є дискретизоване $M \times N$ пікселів і квантоване з точністю K біт на піксель монохромне зображення. Отже, для зберігання цього зображення необхідно $M \times N \times K$ біт інформації.

Якщо припустити, що квантовані значення яскравості не рівноїмовірні, то зменшення інформації можливо шляхом зміни кількості біт інформації для кодування пікселів: більше ймовірні кодуються словами з меншою кількістю біт, менш імовірні – з більшою. Цей метод називається кодуванням словами змінної довжини або ентропійним кодуванням.

Нехай квантований рівень яскравості z має ймовірність $P(z)$ і йому привласнюється слово – код довжини $L(z)$ біт. Тоді середня довжина коду для всього зображення складе $\hat{L} = \sum_b L(z) \cdot P(z)$ біт на піксель. Нижня границя для $\hat{L}$ визначається інформаційною теоремою й називається ентропією випадкової величини:

$$H(f) = -\sum_z P(z) \cdot \log_2 P(z) \le \hat{L}.$$

Таким чином, ентропія – це міра кількості інформації, що несе випадкова величина рівня яскравості z.

$H(z) \ge 0$, оскільки $P(z) \in [0,1]$. З формули $H(f)$ випливає, що чим більше нерівномірний розподіл $P(z)$, тим менше ентропія й тем ефективніше може бути ентропійне кодування.

Найбільш відомі методи ефективного кодування символів засновані на знанні частоти кожного символу присутнього в повідомленні. Знаючи ці частоти, будують таблицю кодів, що володіє наступними властивостями:

– різні коди можуть мати різну кількість біт;

– коди символів з більшою частотою зустрічальності, мають більше біт, ніж коди символів з меншою частотою;

– хоча коди мають різну бітову довжину, вони можуть бути відновлені єдиним образом.

Цими властивостями володіє відомий алгоритм Хаффмана [6].

Візуальна надмірність зображень

Усунення візуальної надмірності зображень є основним резервом скорочення переданої інформації. Для оптимізації процесу кодування з погляду забезпечення передачі найменшого обсягу інформації необхідно, з одного боку, не передавати надлишкову інформацію, а, з іншого, – не допустити надмірної втрати якості зображення.

Дотепер не існує простої і адекватної моделі візуального сприйняття зображень, придатної для оптимізації їхнього кодування [5].

Про кодування кольорових зображень

Більшість кольорових растрових зображень для збереження деталей кольору пікселя застосовують систему кольорів *RGB,* насамперед тому, що ця система використовується для зображення кольорів монітором комп'ютера. Для точного стиску зображення із кольорами *RGB* необхідно всі три компоненти стискати з однаковим рівнем точності. Комбінація цих трьох компонентів визначає відносну яскравість пікселя, а також його колір. Тому зміна кожного із трьох значень позначиться й на яскравості, і на кольорі пікселя.

Так як око людини більше чутливе до зміни яскравості, ніж до зміни кольору, часто виникає бажання перетворити систему кольорів *RGB* у таку систему, де інформація про яскравість пікселів запам'ятовувалася б окремо від інформації про колір. Загальновживаною системою кольорів, що зберігає яскравість пікселя у вигляді окремого компонента даних про його колір, є система *HSB* (тон, насиченість, яскравість) і *YUV* (*Y* – компонента яскравості; *U* і *V* зберігають характеристики кольору).

Огляд деяких алгоритмів стиску із втратами

Метод усіченого блокового кодування (УБК)

Назва методу відбиває той факт, що зображення розбивається на невеликі прямокутні шматки однакового розміру, називані блоками. Цей метод на відміну від більшості інших підбудовує параметри кодування не під деяку усереднену характеристику всього зображення, а під локальні особливості в межах кожного

блоку. Це дозволяє зберегти дрібні деталі зображень. Метод не приводить до розмивання границь, що характерно для деяких інших алгоритмів. Метод УБК зіставимий з більшістю інших методів по ефективності стиску даних і по обсягу обчислень, необхідних для кодування, але не має конкурентів по простоті декодування.

Базовий алгоритм УБК будується в такий спосіб. Зображення, представлене $M \times N$ – матрицею $\|b_{ij}\|$ яскравостей пікселів, розбивається на невеликі прямокутні блоки $m \times n$ елементів. Кожний такий блок обробляється незалежно від інших, тому опишемо алгоритм обробки одного блоку.

Обробка блоку починається з обчислення порога й двох рівнів квантування (описаних нижче), потім проводиться квантування блоку на два рівні, після чого слідує впакування проквантованого блоку. Для визначення рівнів квантування спочатку обчислюються два перших вибіркових моменти – середнє значення C й середній квадрат E:

$$C = \frac{1}{m \times n} \sum_i \sum_i b_{ij} \ , \ E = \frac{1}{m \times n} \sum_i \sum_j b_{ij}^2 ,$$

(де підсумуються елементи зображення в межах блоку) і дисперсія:

$$\sigma^2 = E - c^2 .$$

Гранична величина квантователя d покладається рівною середньому C. Верхній a і нижній b рівні квантування обчислюються по наступних формулах:

$$a = C - \sigma \sqrt{q/(p-q)} \ , \ b = C + \sigma \sqrt{(p-q)/q} ,$$

де $p = m \times n$ – число елементів блоку, q – число елементів блоку, що перевищують поріг d.

Квантування проводиться за правилом:

$$s_{ij} = \begin{cases} a, \text{ якщо } b_{ij} < d \\ b, \text{ якщо } b_{ij} \geq d \end{cases} ,$$

де s_{ij} – елементи зображення після квантування.

Після квантування виходить блок, що містить тільки рівні a й b. Неважко показати, що середнє значення й середній квадрат вихідного й проквантованного блоків збігаються. Практично для зручності наступного впакування замість a записується нуль, замість b – одиниця. Рівні a й b записуються окремо.

Упакування полягає в тому, що блок, що містить тільки нулі й одиниці, інтерпретується як двійкове число, що має $m \times n$ розрядів. Відновлення закодованого зображення також проводиться поблочно й складається в розпакуванні й зворотній підстановці.

З алгоритму видно, що ступінь стиску безпосередньо залежить від розмірів блоку. Найбільш задовільні результати, як по ступені стиску, так і по якості відновленого зображення минулого отримані при використанні блоків розміром 4×4 [3].

Описаний вище спосіб визначення порога й рівнів квантування не є єдиним. Існує ряд інших критеріїв. Важливо, щоб критерій відповідав цілям наступної обробки зображення і її конкретних особливостей.

JPEG

JPEG – один з найпоширеніших і досить потужних алгоритмів, являє собою метод стиску зображень, реалізований різними способами. Працює він як на чорно-білих, так і на повнокольорових зображеннях.

Коефіцієнт архівації в JPEG може змінюватися в межах від 2 до 200 разів. Як і в будь-якого іншого алгоритму стиску із втратами, в JPEG свої особливості. Найбільш відомі 'ефект Гіббса' і дроблення зображення на квадрати. Перший проявляється біля різких границь предмета, створюючи навколо своєрідний 'ореол'. Розбивка на квадрати відбувається, коли задається занадто великий коефіцієнт стиску для даної картинки. Проте, не дивлячись на ці недоліки, для архівації зображень, призначених для перегляду людиною, вона на даний момент є кращим.

Широке застосування JPEG стримується, мабуть, лише тим, що він оперує 24-бітними зображеннями. Тому для того, щоб із прийнятною якістю подивитися

картинку на звичайному моніторі в 256-кольоровій палітрі, потрібне застосування відповідних алгоритмів і, отже, певний час. У додатках, таких, наприклад, як ігри, подібні затримки неприйнятні. Крім того, якщо навні зображення, допустимо, в 8-бітному форматі *GIF* перевести в 24-бітний JPEG, а потім назад в *GIF* для перегляду, то втрата якості відбудеться двічі при обох перетвореннях. Проте, виграш у розмірах архівів найчастіше настільки великий (3-20 разів), а втрати якості настільки малі, що зберігання зображень в JPEG виявляється дуже ефективним.

Перетворення кольорів RGB у кольори YUV

У стиску JPEG застосовується система кольорів *YUV*. Поділ даних *RGB* на дані *YUV* дозволяє програмі стиску приділяти більше увагу даним про яскравість *(Y)*, чим даним про колір *(UV)*. Цей процес називається підвибіркою, так як три компоненти вибираються з різною частотою. Так метод підвибірки, що називається *YUV411*, на кожну вибірку кольору робить чотири вибірки даних про яскравість. Наприклад, якщо рисунок не піддавався підвибірці, те величини *YUV* зустрічаються в ньому з однаковою частотою. При використанні підвибірки *YUV411* на шість значень вибірки обробленого файлу доводиться дванадцять значень вибірки вихідного файлу. Таким чином, підвибірка даних відразу зменшує розмір файлу зображення.

Метод стиску JPEG

Процес стиску за схемою JPEG складається із трьох кроків (не вважаючи підвибірку). Перший крок – це запис зміни значень пікселів у вигляді зміни частот: як швидко міняються яскравість і колір пікселів. Другий крок – угруповання цих окремих змін частот за середнім значенням (перший етап стиску). І третій етап – стиск цих усереднених даних за допомогою модифікованого алгоритму кодування Хаффмана.

Зміна частоти

JPEG визначає зміну частоти даних за допомогою дискретного перетворення Фур'є (ДПФ), що застосовується для кожної піксельной

компоненти в обраній області пікселів. Наприклад, вибирається група з 8×8 пікселів, і до неї застосовується перетворення ДПФ: спочатку до величин червоних компонентів у всій групі, потім зелених, і, нарешті, синіх.

Замість дійсних значень пікселів величини ДПФ зберігають швидкість зміни інтенсивності від пікселя до пікселю. Насправді даних про частоту виходить більше, ніж вихідних піксельних даних, але наступні два кроки це усувають.

Усереднення

Після обчислення за допомогою ДПФ значень зміни частоти ці величини усередняються відповідно до плаваючої шкали відносної важливості. Це значить, що зміни частоти, які менше впливають на загальний вид зображення (наприклад, швидкі зміни частоти), усередняються більше інших значень. Саме на цій стадії стиску зображення відбуваються втрати, так як величина застосовуваного усереднення може регулюватися.

Стиск методом Хаффмана

Остаточно усереднені дані про частоту стискуються за допомогою модифікованого алгоритму кодування Хаффмана, що особливо ефективний для цього типу даних, так як будує таблиці кодів, що базуються на частоті повторення величин.

Фрактальний стиск

Ця група алгоритмів є самою перспективною й розвивається найбільше бурхливо. Основні ідеї фрактального стиску належать Barnsley [7]. Перші практичні результати були отримані Jacquin [8] в 1992 році. Алгоритм ґрунтується на ідеї подоби між елементами зображення. Коефіцієнти стиску у фрактальних алгоритмів варіюються в межах 2-2000 разів. Причому більші коефіцієнти досягаються на реальних зображеннях, що нетипово для попередніх алгоритмів. Крім того, при розархівації зображення можна масштабувати. Унікальна особливість цього алгоритму полягає в тому, що збільшене зображення не дробиться на квадрати.

Недоліком цього алгоритму є потреба в великих обчислювальних потужностях при архівації. При цьому розпакування вимагає менше обчислень, ніж в JPEG. Фактично це перший істотно несиметричний алгоритм. Причому, якщо у всіх попередніх алгоритмів коефіцієнт симетричності (відношення часу архівації вчасно розархівації) не перевищує 3, то у фрактального алгоритму він коливається від 1000 до 10000.

Метод фрактального стиску зображень

Нехай задане зображення $M \times N$ пікселів. Позначимо через $\Omega = (0, M] \times (0, N] \in R^2$ безліч точок зображення. При цьому кожному пікселю (i, j) зіставляється площадка:

$$\{(\xi, \eta) \in \Omega, \xi \in (j-1, j], \eta \in (i-1,]\},$$

точка $(i, j) \in \Omega$ – правий верхній кут площадки пікселя.

Основна ідея фрактального стиску складається в поданні вихідного зображення, заданого функцією $z = f(\xi, \eta)$, за допомогою деякого стискаючого відображення, нерухлива точка якого відповідає вихідному зображенню.

Нехай X – безліч векторів з R^n, де $n = M \times N$ й i-я координата вектора відповідає одному пікселю зображення: $x_i = f(1 + (i-1) \bmod N, \left\lfloor \frac{i}{N} \right\rfloor + 1)$. Тут $i = 1, ..., n$, $\lfloor \rfloor$ – символ округлення донизу.

Потрібне нам відображення будемо шукати у вигляді: $A: X \to X$.

Для завдання такого відображення розіб'ємо Ω на безліч R-блоків $R_1, R_2, ..., R_r$, де $R_i \subseteq \Omega, i = 1, ..., r$ є квадратний $B \times B$ піксельний фрагмент зображення й $\bigcup_i R_i = \Omega$. Розглянемо також безліч D-блоків: $D_1, D_2, ..., D_d$, що де $D_i \subseteq \Omega$ представляють квадратні (можливо пересічні) $2B \times 2B$ піксельні фрагменти й $\bigcup_i D_i = \Omega$.

Процес кодування зображення полягає в пошуку двох сукупностей аффіних перетворень W_i і F_i.

$W_i : D_{j(i)} \to R_i, i = 1,...,r$ – біекція, дія якої описується в такий спосіб: нехай $(\xi^{©}, \eta^{©}) = W_i(\xi, \eta)$ має вигляд:

$$\begin{bmatrix} \xi' \\ \eta' \end{bmatrix} = \begin{bmatrix} \alpha_i & \beta_i \\ \gamma_i & \delta_i \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix} + \begin{bmatrix} \varphi_i \\ \psi_i \end{bmatrix}.$$

Очевидно, існує всього вісім способів такого перетворення квадрата у квадрат.

Функціонал $F_i : R_i \to R,\ i = 1,...,r$ задає колірні характеристики кожного пікселя (ξ, η) блоку R_i й визначається як усереднені значення яскравості пікселів прообразу (ξ, η):

$$F_i(\xi, \eta) = s_i \cdot \frac{1}{4} \sum \left(f(\tilde{\xi}, \tilde{\eta}) : (\tilde{\xi}, \tilde{\eta}) \in W^{-1}((\xi - 1, \xi] \times (\eta - 1, \eta]) \mid Z^2 \right) + o_i, \quad (3.1)$$

де s_i відповідає за контрастність, а o_i за яскравість.

Представимо шукане відображення A, як результат дії сукупностей W_i і F_i в такий спосіб: нехай координати вектора $x' = A(x)$ мають вигляд $x_k^{©} = F_{i(k)}(\xi(k), \eta(k))$, де $\xi(k) = 1 + (k-1) \bmod N$, $\eta(k) = \left\lfloor \frac{k}{N} \right\rfloor + 1$ – координати правого верхнього кута площадки k-го пікселя, а $i(k)$ – номер R блоку, що містить точку $(\xi(k), \eta(k))$.

Обґрунтування методу фрактального стиску

При фрактальному кодуванні відображення A підбирається таким чином, щоб мінімізувати відстань між вектором x і його образом $A(x)$ у деякій метриці.

Відповідно до теореми Банаха, якщо A – стискаюче відображення й (X, ρ) – повний метричний простір з метрикою ρ, тоді послідовність $\{x_k\}$, побудована за правилом:

$$x_{k+1} = A(x_k), \quad (3.2)$$

сходиться для довільної точки $x_0 \in X$ до єдиної нерухливої точки $x_A \in X$ перетворення A:

$$x_A = A(x_A). \quad (3.3).$$

Тому для відновлення зображення необхідно запам'ятати лише відображення A.

Неважко бачити, що якщо сімейство відображень F_i таке, що:

$$L = \max_i |s_i| < 1, \text{ те } \rho(x,y) \le L \cdot \rho(A(x), A(y)), \quad (4)$$

де ρ метрика, породжена нормою l^∞. Таким чином, для збіжності ітераційного процесу (3.2) досить контролювати параметр s_i, щоб $s_i < 1 \forall i = 1,...,r$. Проте, необхідно відзначити, що умова $L < 1$ не є необхідним для збіжності процесу відновлення.

Загалом кажучи x_A не завжди буде точно збігатися з образом $A(x)$, але, при виконанні деяких умов на оператор A можна гарантувати, що x_A буде «майже» також близько до x, як і Ax.

Відповідну оцінку відстані між x і $A(x)$ дає наступна теорема (Collage theorem) [9].

Теорема: $\rho(x, x_A) \le \frac{1}{1-L} \rho(x, A(x))$, де $L < 1$ – константа Ліпшица стискаючого оператора A в деякій метриці ρ.

У випадку використання евклідової метрики константа Ліпшица оператора A може бути обчислена в такий спосіб [10]

$$L = \max_{1 \le j \le d} \sqrt{\frac{1}{4} \sum \left(s_i^2 \ : \ j(i) = j \right)} \ . \quad (3.5)$$

У пропонованому далі алгоритмі як критерій близькості $A(x)$ до вихідного вектора зображення x використовується евклідова метрика. А саме, потрібно знайти таке відображення A, що $\rho(A(x), x)$ мінімально. Це рівносильно завданню: знайти для кожного R-блоку R_i, $i = 1,...,r$ таку трійку $(W_i, D_{j(i)}, F_i)$, що:

$$\sigma^2_{R_i}(W_i, D_{j(i)}, F_i) = \sum \left(\left[f(\xi,\eta) - F_i(\xi,\eta) \right]^2 : (\xi,\eta) \in R_i \mid Z^2 \right) \to \min, \quad (3.6)$$

де підсумовування ведеться по всім пікселям блоку R_i.

При рішенні цього завдання для будь-яких варіантів $D_{j(i)}$ і W_i можна підібрати оптимальні параметри s_i й o_i, використовуючи метод найменших квадратів для лінійної регресії.

Нехай дані дві послідовності зі B^2 значень кольорів пікселів до й після застосування W_i:

$$z_1 = f(\xi_1,\eta_1), z_2 f(\xi_2,\eta_2),..., z_k = f(\xi_{B^2},\eta_{B^2})$$

та

$$z_1^{'} = \frac{1}{4}\sum\left(f(\tilde{\xi},\tilde{\eta}) : (\tilde{\xi},\tilde{\eta}) \in W^{-1}((\xi_1 - 1,\xi_1] \times (\eta_1 - 1,\eta_1]) \mid Z^2 \right),$$

…,

$$z_k^{'} = \frac{1}{4}\sum\left(f(\tilde{\xi},\tilde{\eta}) : (\tilde{\xi},\tilde{\eta}) \in W^{-1}((\xi_k - 1,\xi_k] \times (\eta_k - 1,\eta_k]) \mid Z^2 \right),$$

де $(\xi_k,\eta_k) \in R_i$ й $z_k^{'}$ усереднені значення яскравості пікселів прообразу (ξ_k,η_k) $\forall k = 1,..., B^2$.

Ми можемо знайти s_i й o_i для (3.1) мінімізувавши суму:

$$\sigma^2_{R_i} = \frac{1}{B^2}\sum_{k=1}^{B^2}(s_i z_k^{'} + o_i - z_k)^2. \qquad (3.7)$$

Функція (3.7) має мінімум у тих точках, у яких частні похідні від $\sigma^2_{R_i}$ по параметрах s_i і o_i звертаються в нуль. У результаті диференціювання й елементарних перетворень для визначення параметрів одержуємо систему двох лінійних рівнянь із двома невідомими s_i й o_i звідки одержуємо, що:

$$s_i = \frac{\left[B^2 \cdot \left(\sum_{k=1}^{B^2} z_k' z_k \right) - \left(\sum_{k=1}^{B^2} z_k' \right) \cdot \left(\sum_{k=1}^{B^2} z_k \right) \right]}{B^2 \cdot \sum_{k=1}^{B^2} z_k'^{\,2} - \left(\sum_{k=1}^{B^2} z_k' \right)^2}, \qquad o_i = \frac{1}{B^2} \left(\sum_{k=1}^{B^2} z_k - s_i \cdot \sum_{k=1}^{B_2} z_k' \right).$$

Зокрема, якщо $s_i = 0$, те $o_i = \frac{1}{B^2} \sum_{k=1}^{B^2} z_k$. У такий спосіб середньоквадратична помилка перетвореного блоку від його вихідного стану

$$\sigma_{R_i}^2 = \frac{1}{B^2} \left[\sum_{k=1}^{B^2} z_k^2 + s \cdot \left(s \cdot \sum_{k=1}^{B^2} z_k'^{\,2} - 2 \cdot \sum_{k=1}^{B^2} z_k' z_k + 2 \cdot o \sum_{k=1}^{B^2} z_k' \right) + o \cdot \left(o \cdot B^2 - 2 \cdot \sum_{k=1}^{B^2} z_k \right) \right].$$

Завдання пошуку D-блоку $D_{j(i)}$ і його аффіного відображення W_i може бути вирішена різними способами. Наприклад за допомогою ГА, алгоритм Фішера [10], повний перебір і т.д.

Алгоритми пошуку фрактальної моделі зображення

Алгоритм Фішера

Ідея алгоритму полягає в тому, щоб деяким чином класифікувати D-блоки й R-блоки, а пошук близького D-блоку робити в тому же класі, до якого відноситься рангова область. Робиться це в такий спосіб.

Вихідні блоки розбиваються на чотири частини. Для кожної із частин підраховується середнє значення A_i й дисперсія V_i пікселів. Далі блоки класифікуються по наступному принципі. Визначимо три базових типи блоків:

– тип 1: $A_1 \geq A_2 \geq A_3 \leq A_4$;

– тип 2: $A_1 \geq A_2 \geq A_4 \geq A_3$;

– тип 3: $A_1 \geq A_4 \geq A_2 \geq A_3$.

Зрозуміло, що будь-який блок за допомогою відповідного аффіного перетворення квадрата у квадрат можна привести до виду, що відповідає одному із зазначених типів. Після того, як ми зафіксували три основних класи, блоки

класифікуються по дисперсії. Таким чином, у кожному із трьох класів з'являються 24 підкласи, разом 72 класу. Пошук близького до R-блоку D-блоку виробляється перебором у відповідному класі.

Генетичний алгоритм

Генетичний алгоритм (ГА) являє собою алгоритмічний підхід до рішення екстремальних завдань однокритеріального вибору, заснований на моделюванні основних факторів еволюційного розвитку популяції.

При використанні ГА для пошуку оптимальних рішень кожний елемент $x \in X$ простору оптимізації повинен бути представлений як вектор $b \in B$ з N символів двійкового алфавіту A = {0,1}, де $B = A^N$. Необхідно також, щоб простір оптимізації X складалося з кінцевого числа елементів.

Популяцією $П = (\chi^1, \chi^2, \ldots, \chi^M)$чисельності M уважається вектор простору B^M, координати якого називаються генотипами осіб даної популяції.

Кроком ГА є перехід від поточного покоління до наступного, тобто одержання нової популяції $П_{t+1}$ з $П_t$. У побудові чергової особи нової популяції беруть участь оператори кроссинговера, мутації й випадковий оператор відбору, *Select*: $B^M \to \{1,\ldots,M\}$ дія якого складається у виборі номера особи батька при породженні чергового нащадка.

Для визначення ГА в необхідно задати оператор кроссинговера (схрещування) *Cross*: $B \times B \to B \times B$ і оператор мутації *Mut*: $B \to B$.

Дія кроссинговера $(\chi', \tau') = Cross(\chi, \tau)$ полягає у виборі випадковим образом деякої позиції *j*, рівномірно розподіленої від *1* до N-1, після чого результат формується у вигляді:

$$\chi' = (\chi_1, \chi_2, ..., \chi_j, \tau_{j+1}, ..., \tau_N),\ \tau' = (\tau_1, \tau_2, ..., \tau_j, \chi_{j+1}, ..., \chi_N).$$

Вплив кроссинговера регулюють за допомогою ймовірності P_{Cross} спрацьовування цього оператора (у противному випадку все залишається без змін).

Оператор мутації в кожній позиції аргументу із заданою ймовірністю P_{mut} заміняє її вміст на випадковий елемент двійкового алфавіту A, обраний відповідно до рівномірного розподілу (у противному випадку все залишається без змін).

Цільова функція вихідного завдання, заміняється в ГА на ненегативну функцію придатності генотипу $\Phi(\chi)$, де $\chi \in \mathrm{B}$.

Процес роботи алгоритму являє собою послідовну зміну поколінь, на кожному кроці якої популяція Π_{t+1} наповнюється парами нащадків від осіб популяції Π_t за формулою:

$$(\chi_k^{t+1}, \chi_{k+1}^{t+1}) = Mut(Cross(\chi_{Select(\Pi_t)}^{t}, \chi_{Select(\Pi_t)}^{t})),$$

де $(\chi_k^{t+1}, \chi_{k+1}^{t+1})$ – особи з найменшою придатністю популяції Π_t. Тобто індивіди витягають попарно з Π_t і після кроссинговера й мутації містяться в. Π_{t+1} Зміну ймовірностей мутації й кроссинговера дозволяє регулювати роботу ГА й набудовувати його на конкретні завдання.

Модифікація генетичного алгоритму для завдання фрактального стиску

Опишемо схему ГА в застосуванні до завдання фрактального стиску. Як генотип ГА зручно взяти вектор, компонентами якого будуть піксельні координати області $D_{j(i)}$ вихідного зображення, певного на тороїдальній поверхні, і число що кодує аффіне перетворення W_i. Є вісім способів аффіного перетворення квадрата у квадрат: поворот на чотири сторони або дзеркальне відбиття й поворот на чотири сторони. Отже, на кодування цього перетворення досить трьох біт. Функцію придатності покладемо рівною:

$$\Phi = \frac{1}{1 + \sum\left(\left[f(\xi,\eta) - F_i(\xi,\eta)\right]^2 : (\xi,\eta) \in R_i \mid Z^2\right)},$$

де в нижній частині під знаком суми – евклідову відстань між вихідним і перетвореним блоком. Дана функція задовольняє вимоги ГА (ненегативна) і адекватна для оператора рулеточної селекції, при якій кожний індивід $\chi^{i,t}$

популяції $П_t$ виявляється батьком при формуванні чергової особи $\chi^{i,t+1}$ популяції $П_{t+1}$ з імовірністю:

$$Pselect(\chi^{i,t}) = \frac{\Phi(\chi^{i,t})}{\sum_j \Phi(\chi^{j,t})}.$$

При такому поданні хромосом, що визначають даний генотип, будь-який вектор простору рішень завжди допустимий й має ненульову придатність.

Оператор мутації для даного алгоритму – стандартний, а оператор кроссинговера був модифікований у такий спосіб:

– позиція кроссинговера може розташовуватися тільки в місцях стикування двійкового подання координат.

– виконується тільки для близькородственних осіб, тобто, якщо відстань $\rho(\chi,\tau)$ мало, де ρ – деяка метрика.

Збільшення швидкодії методу

Як уже було сказано вище, недоліком даного методу стиску є дуже великий обсяг обчислень для кодування зображень. Для програм що реалізують даний метод, застосовуються як звичайні методи збільшення швидкості програм: цілочисельна арифметика, ассемблерна реалізація й інші, так і деякі специфічні прийоми: використання аффіних перетворень тільки одного типу, різні варіанти розбивки зображення на області R_i.

Приведемо схеми двох алгоритмів, які для деяких класів зображень можуть значно зменшити обсяг обчислень. Параметрами першого алгоритму служать рівень втрат при кодуванні й мінімальний розмір областей R_i. Цей алгоритм забезпечує рівномірну якість кодування всього зображення. Параметром другого алгоритму є кількість областей R_i, використовуваних для кодування зображення, що прямо впливає на обсяг обчислень, але він не забезпечує достатньої точності кодування окремих фрагментів зображення.

Алгоритм 1

– Виберемо припустимий рівень втрат при кодуванні e.

– $R_1 = \Omega$ і позначимо його як неопрацьований фрагмент.

– Поки є неопрацьований фрагмент R_i виконувати:

1. Знайти $D_{j(i)}$, W_i і F_i, які щонайкраще наближають R_i (на яких досягається мінімум $\sigma^2_{R_i}(W_i, D_{j(i)}, F_i)$).

2. Якщо $\sigma^2_{R_i}(W_i, D_{j(i)}, F_i) < e$ або розмір $R_i < \min$, то позначити R_i як оброблене. Інакше, розбити R_i на більше дрібні фрагменти й позначити їх як неопрацьовані.

Алгоритм 2

– Виберемо максимальне число N фрагментів R_i.

– Додамо фрагмент $R_1 = \Omega$ у список перетворень і позначимо його як неопрацьований.

– Поки є неопрацьовані фрагменти в списку виконувати:

1. Для кожного неопрацьованого фрагмента знайти відповідні $D_{j(i)}$, W_i і F_i.

2. Знайти в списку фрагмент R_i найбільшого розміру й найбільшою метрикою $\sigma^2_{R_i}(W_i, D_{j(i)}, F_i)$.

3. Якщо число фрагментів у списку менше N, тоді розбити фрагмент R_i на більше дрібні й занести їх у список як неопрацьовані. Викреслити з писку фрагмент R_i.

Обчислювальний експеримент

Описаний метод фрактального стиску статичних зображень був запрограмований із застосуванням об’єктно-орієнтованого підходу. Програма працює як із чорно-білими, так і з повнокольоровими зображеннями формату *Windows Bitmap*.

Для оцінки якості закодованого 8-бітного зображення або одного каналу кольорового використовувалося співвідношення сигнал-шум, вимірюване в децибелах (d) і визначене як:

$$PSNR = 10 \cdot \lg\left(\frac{255^2}{\sum (f(\xi,\eta) - \tilde{f}(\xi,\eta))^2}\right),$$

де підсумовування ведеться по всім пікселям вихідного f й наближеного $\tilde{f}$ зображення.

Алгоритм кодування реалізований у двох варіантах.

1. Алгоритм кодування з фіксованим розміром блоків.

Алгоритм оперує із прямокутними областями однакового розміру. Розмір областей фіксований від початку роботи алгоритму до кінця.

Перевагою даного алгоритму полягає в тому, що при відповідному виборі розмірів оброблюваних областей забезпечується рівномірна якість кодування всього зображення. Недоліком алгоритму є малий коефіцієнт стиску.

2. Алгоритм кодування з розбивкою зображення на дерево підблоків.

За основу був узятий *алгоритм 1* з попереднього параграфа. При недостатній точності кодування оброблюваний фрагмент розбивається на чотири частини, кожна з яких обробляється так, як і всі інші.

Дана розбивка ефективно з погляду зберігання його у файлі. Так будь-яку таку розбивку можна представити у вигляді двійкової послідовності символів, де 1 – означає те, що блок розбитий, 0 – інакше. При тестуванні алгоритм показав кращі результати, ніж його попередник, по ступеню стиску, але не завжди забезпечує достатню точність кодування деяких однотонних областей.

Ступінь стиску зображення при фрактальному кодуванні залежить від розмірів оброблюваних R-блоків, параметрів s_i і o_i перетворення F_i. Для чорно-білих зображень вона може бути обчислена за формулою:

$$\frac{2^8 * \textit{розмір блоку у пікселях}}{\textit{число біт для } s + \textit{число біт для } o + \lceil \log_2(8 \times \textit{розмір кодової книги}) \rceil},$$

де $\lceil\ \rceil$ – символ округлення нагору. Таким чином, ми можемо одержати додатковий ступінь стиску, якщо квантувати параметри s_i й o_i. Очевидно, що після квантування цих параметрів середньоквадратичне відхилення між вихідним і наближеним зображенням зросте на величину не більшу, ніж

$$\left(\frac{\max\limits_i(s_i) - \min\limits_i(s_i)}{2^{\textit{число біт для } s}} * 2^8 + \frac{\max\limits_i(o_i) - \min\limits_i(o_i)}{2^{\textit{число біт для } o}} \right)^2.$$

Пошук фрактальної моделі (для кожного R-блоку відповідний D-блок і перетворення *W*) зображення здійснювався за допомогою модифікованого генетичного алгоритму, описаного раніше. При проведенні обчислювального експерименту ГА припиняв свою роботу з витікання заданого числа поколінь або при досягненні певної якості кодування, тобто якщо ступінь пристосованості найкращої особи буде вище деякого заздалегідь заданого числа. Кращі результати були отримані при запуску ГА з наступними параметрами $P_{Mut} = 0.1$, $P_{Cross} = 0.75$.

Слід зазначити, що використання генетичних алгоритмів для завдання фрактального стиску є одним з варіантів відходу від повного перебору. Так для зображення розміром 256×256 пікселів, для кожного R-блоку потрібно перебрати $2^8 * 2^8 * 2^3$ комбінацій D-блоків з аффіним перетворенням W. Наступна діаграма показує середнє число перелічених варіантів при роботі ГА залежно від числа поколінь і розміру популяції.

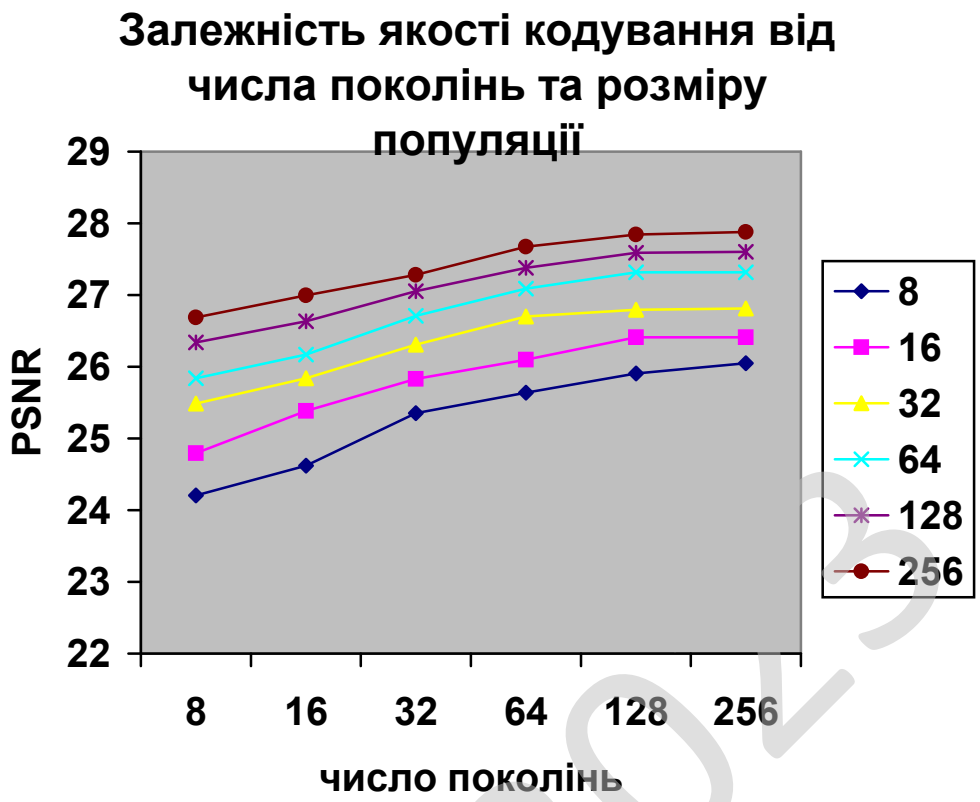

Рисунок 3.1 – Результати застосування генетичних алгоритмів для фрактального стиску

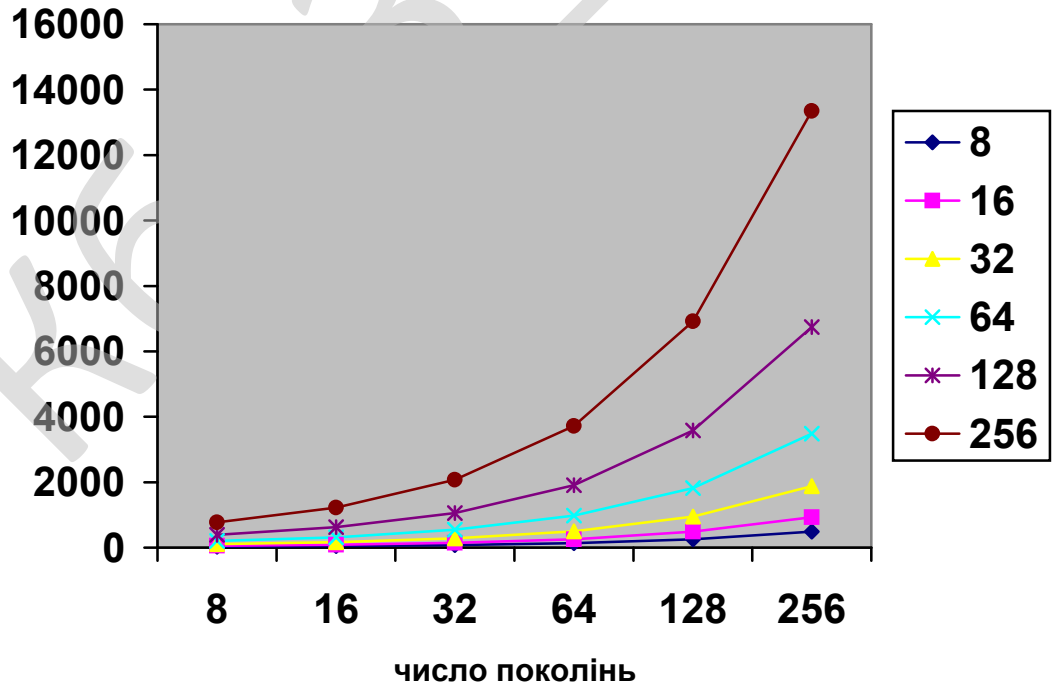

Рисунок 3.2 – Середнє число перелічених варіантів при роботі ГА

Висновок:

– Із цих графіків видно, що запропонований ГА має значні перевагу в порівнянні з алгоритмом повного перебору за часом роботи.

– Представляється доцільним використання даного алгоритму в сполученні з іншими евристиками такими, як алгоритм класифікації Фішера, пошук з обліком фрактальної розмірності й т.д.

– Наведений обчислювальний експеримент показує, що ГА може бути використаний для стиску графічних зображень. Надалі було б цікаво зрівняти ефективність ГА з іншими алгоритмами пошуку перетворення, що кодує.

3.2 Розробка структурної схеми

Структурна схема системи наведена на рисунку 3.3. 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.

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.

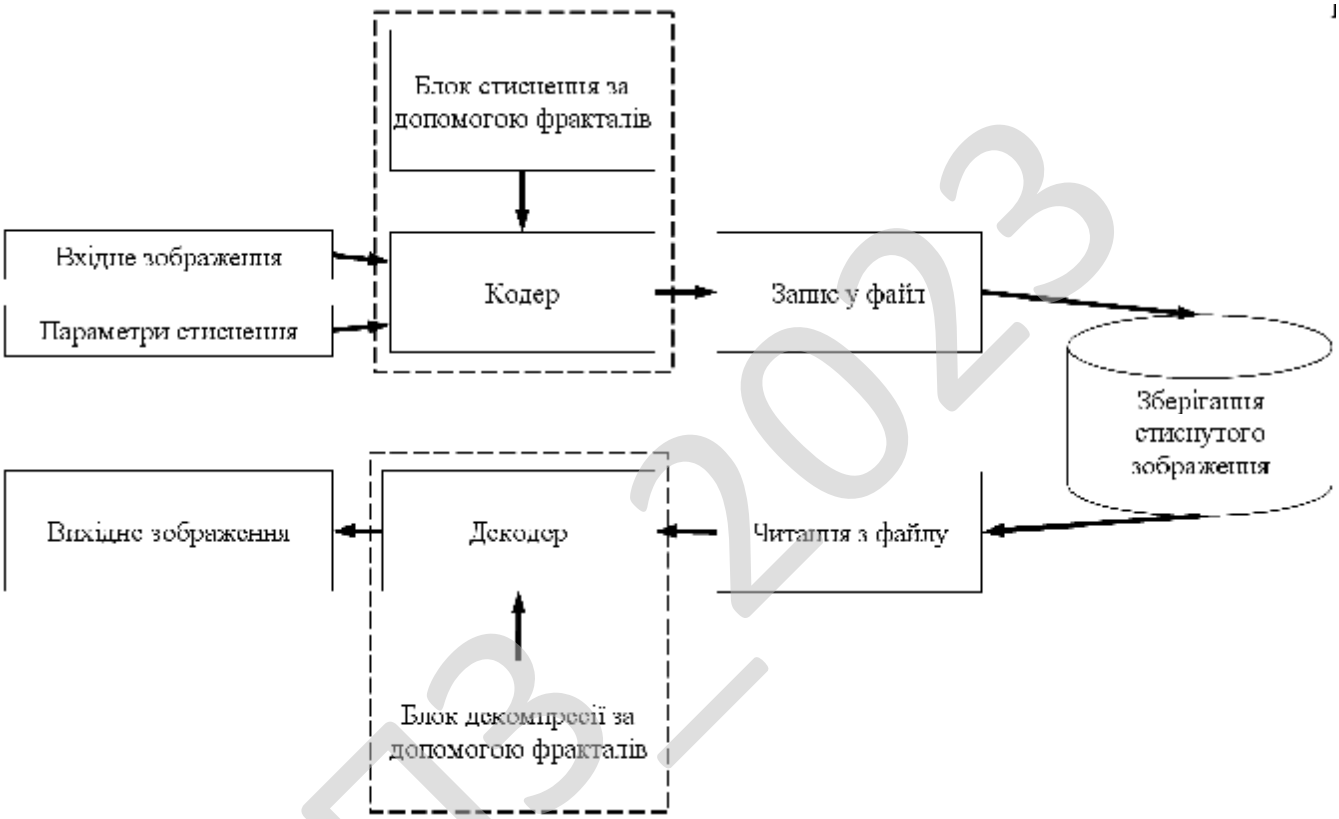

Рисунок 3.3 – Структурна схема системи

Після робіт Бенуа Мандельброта математики продовжували пошук підстав для застосування фрактальної геометрії. Так, Джон Хатчинсон в 1981 році розробив апарат теорії ітерованих функцій. Пізніше Майкл Барнсли написав книгу «Фракталы всюди» (*Fr**A**ct**A**ls Ever**yw**here*, 1988), що є однієї із ключових робіт із фрактальному стиску. У книзі викладені математичні основи систем ітеруємих функцій, головним результатом є теорема колажу (яка пояснює, які умови необхідні для відтворення зображення).

Фрактальна математика (пряма завдання) добре підходить для моделювання ландшафтів. Зворотне завдання полягає в тім, щоб застосувати системи ітеруємих функцій для стиску зображень.

Еволюція розвитку алгоритмів фрактального кодування складається із двох частин: класична (Майкл Барнсли й Алан Слоун) і сучасна (Майкл Барнсли й Арно Жакан (A. Jaquin)).

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

$$f : I^2 \to \{1,2,...,N\}, \tag{3.8}$$

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:

$$d_2(f,g) = \left(\int_{I^2} |f(x,y) - g(x,y)|^2 dxdy \right)^{1/2}. \tag{3.9}$$

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:

$$d_{rms} = \left[\sum_{i=1}^{n} \sum_{j=1}^{m} \left| f(x_i, y_j) - g(x_i, y_j) \right|^2 \right]^{1/2}. \qquad (3.10)$$

Цифрові зображення являють собою матрицю фіксованих значень функції $f(x,y)$, узятих у фіксованих точках $f(x_i,y_j)$. Наведене співвідношення (3.10) 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:

$$\overline{w_i}(x, y) = A_i \begin{pmatrix} x \\ y \end{pmatrix} + b_i . \qquad (3.11)$$

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.

$$\overline{w_i}(f)(x, y) = s_i f(\overline{w_i^{-1}}(x, y)) + o_i , \qquad (3.12)$$

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 (3.12) є базовим для зображень у градаціях сірого, ми його будемо використовувати при стиску.

У випадку використання кусочно-определенной 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:

$$W = \bigcup_{i=1}^{N} w_i, f = x_W . \tag{3.13}$$

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

$$f = W(f) = w_1(f) \cup w_2(f) \cup \dots \cup w_N(f). \tag{3.14}$$

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):

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \quad (3.15)$$

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.

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.

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).

3.3 Розробка функціональної схеми

На рисунку 3.4 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Схема складається з двох великих функціональних блоків:

– Блок компресії.

– Блок декомпресії.

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.

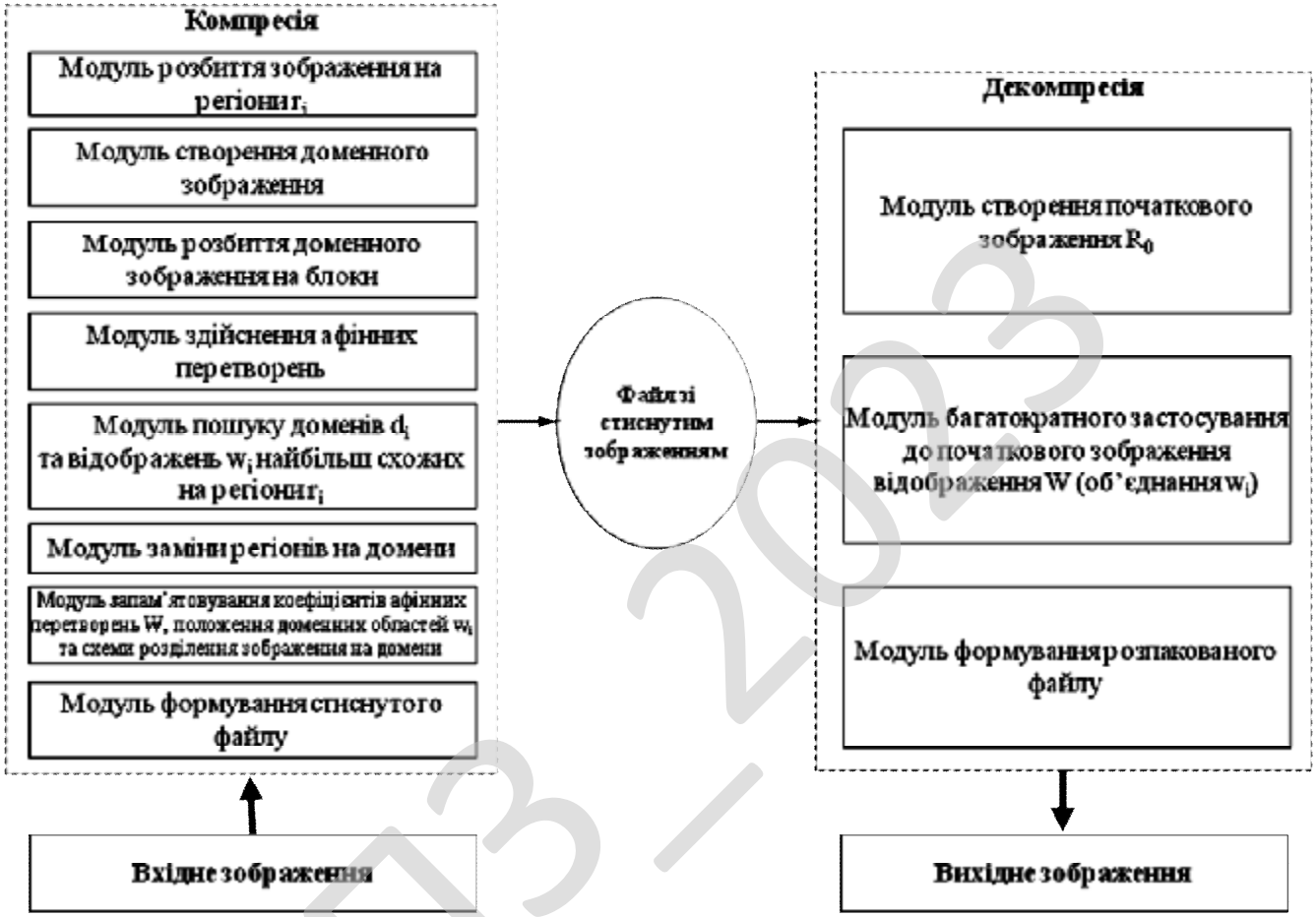

Рисунок 3.4 – Структурна схема системи

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.5. З нього видно, що процеси у системі взаємодіють наступним чином. Першим завантажується процес

виведення головного вікна програми. Він взаємодіє з наступними процесами:

– Процес встановлення параметрів декомпресії.

– Процес відкриття стиснутого файлу.

– Процес відкриття файлу зображення.

– Процес встановлення параметрів компресії.

Процес встановлення параметрів декомпресії взаємодіє з наступними процесами:

– Процес встановлення величини розміру регіону.

– Процес встановлення кількості ітерацій.

Процес відкриття стиснутого файлу взаємодіє з процесом декомпресії файлу.

Процес декомпресії файлу взаємодіє з процесом виведення розпакованого зображення на екран.

Процес виведення розпакованого зображення на екран взаємодіє з процесом вибору формату зображення.

Процес вибору формату зображення взаємодіє з процесом збереження розпакованого файлу у вказаному форматі.

Процес відкриття файлу зображення взаємодіє з наступними процесами:

–Процес компресії файлу.

– Процес виведення зображення на екран.

Процес компресії файлу взаємодіє з наступними процесами:

– Процес збереження стиснутого файлу.

– Процес попереднього перегляду результатів перед збереженням.

Процес попереднього перегляду результатів перед збереженням взаємодіє з процесом декомпресії зображення.

Процес декомпресії зображення взаємодіє з процесом виведення зображення на екран.

Процес встановлення параметрів компресії взаємодіє з наступними процесами:

– Процес встановлення величини зміщення домену.

– Процес встановлення величини розміру регіону.

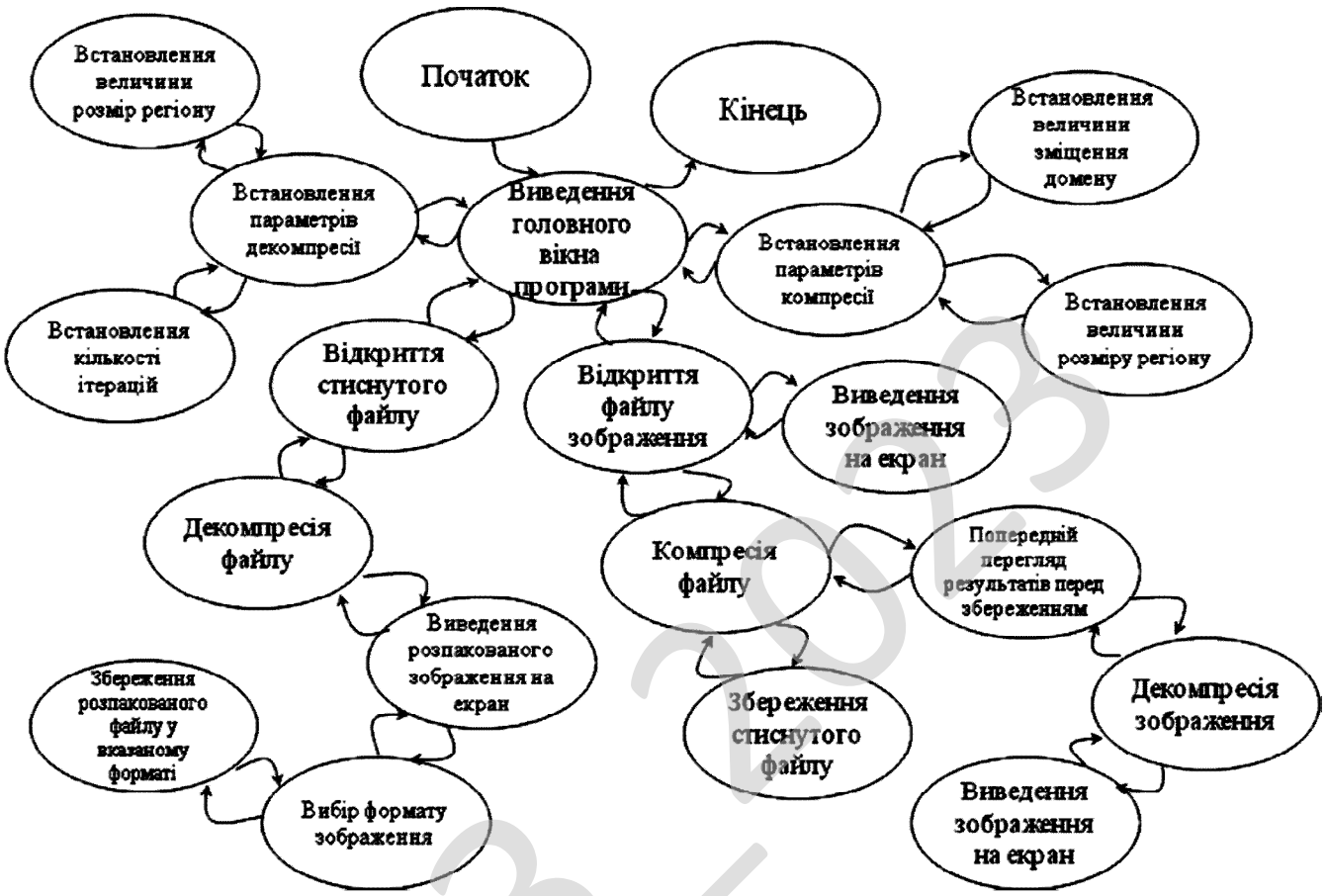

Рисунок 3.5 – Діаграма взаємодії процесів

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

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

4.1 Блок–схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення основного вікна програми. Після цього користувач обирає яку він може зробити:

– Відкрити зображення.

– Стиснути зображення.

– Виконати декомпресію.

Якщо користувач вирішує відкрити зображення, тоді виконуються наступні дії:

– Відкриття файлу із зображенням.

– Виведення зображення на екран.

Якщо користувач вирішує стиснути зображення, тоді виконуються наступні дії:

– Відбувається вибір величини зміщення домену.

– Відбувається вибір величини розміру регіону.

– Запускається підпрограма стиснення зображення.

– Виводиться зображення після стиснення.

– Відбувається запис стиснутого зображення у файл.

Якщо користувач вирішує виконати декомпресію, тоді виконуються наступні дії:

– Запускається підпрограма декомпресії зображення.

– Виводиться зображення після декомпресії.

– Відбувається запис зображення у файл.

Після виконання усіх цих дій, користувач обирає, працювати йому далі з програмою, або ні.

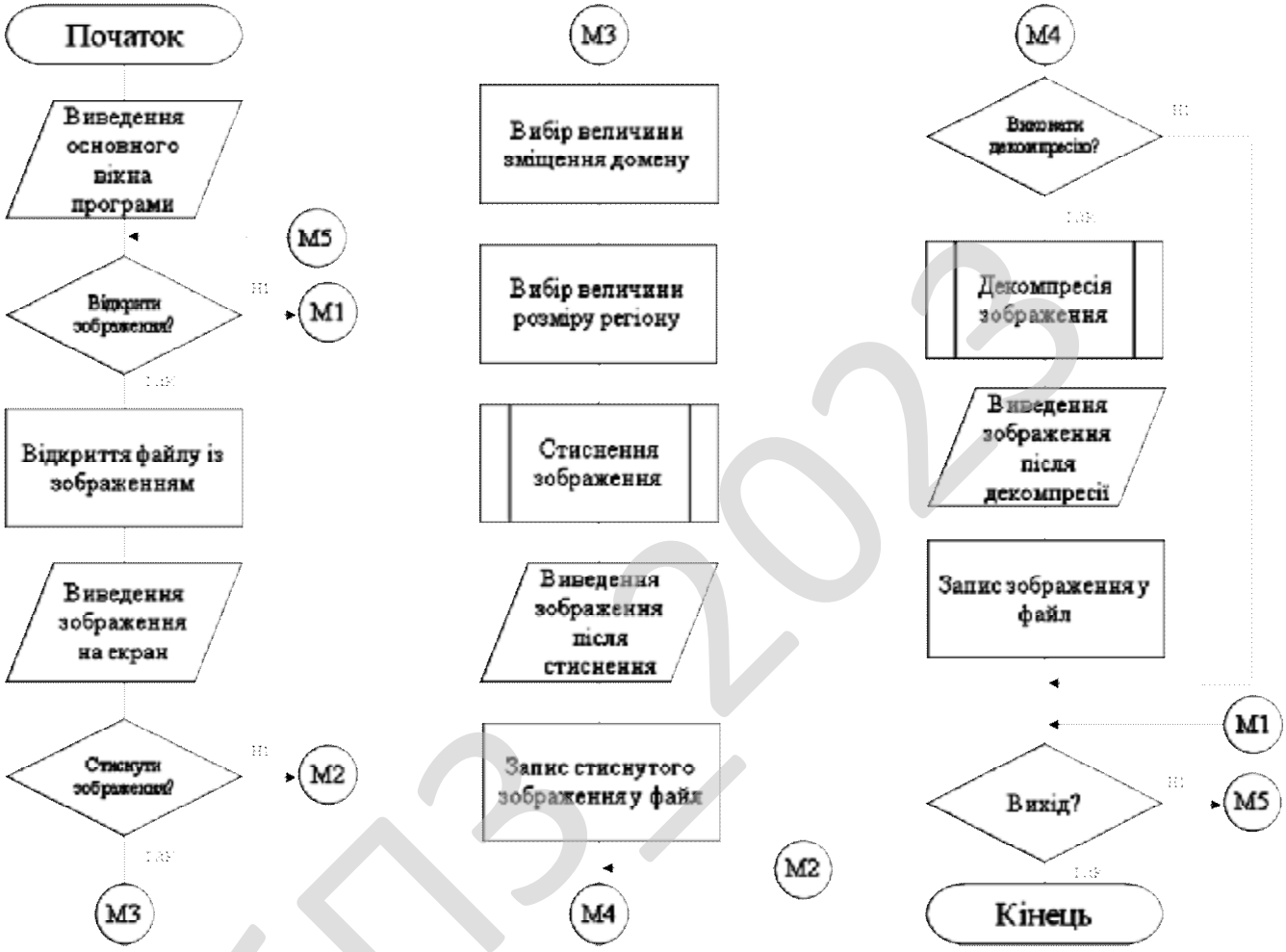

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 зображено блок-схему алгоритму роботи підпрограми стиснення зображення за допомогою фракталів. Вона працює наступним чином. Для роботи підпрограми послідовно виконуються наступні кроки:

– Розбиття зображення на регіони.

– Створення вихідного файлу.

– Запис у вихідний файл кількості регіонів по вертикалі та горизонталі, і розміру регіонів.

– Створення доменного зображення, яке менше основного у 4 рази.

– Визначається кількість регіонів.

– Визначається кількість доменів.

Далі запускається цикл, у якому перебираються усі регіони. У ньому виконуються такі дії:

– Афінне перетворення регіону.

– Запуск підциклу, у якому перебираються усі домени, де виконуються наступні дії: порівняння і-го домену з j-тим регіоном; вибір найбільш схожого домену на і-й регіон.

– Заміна регіону на обраний домен.

– Запис у файл коефіцієнтів аффіних перетворень: номеру домену та номеру афінного перетворення.

– Запис у файл різниці середньої яскравості між регіоном та доменом.

На цьому цикл по доменам, закінчує свою роботу й відбувається закриття вихідного файлу. Після цього підпрограма стиснення зображення за допомогою фракталів завершує свою роботу.

На рисунку 4.3 зображено блок-схему алгоритму роботи підпрограми декомпресії зображення за допомогою фракталів. Вона працює наступним чином.

Спершу створюється початкове зображення.

Після цього користувач обирає, яку декомпресію йому робити: повну, або часткову.

Якщо користувач обирає повну декомпресію, тоді виконуються наступні дії:

– Отримання зображення R_1, застосування до зображення R_0 відображення W.

Якщо $R_0 = R_1$ тоді вважається що декомпресія зроблена.

У противному випадку $R_0 = R_1$.

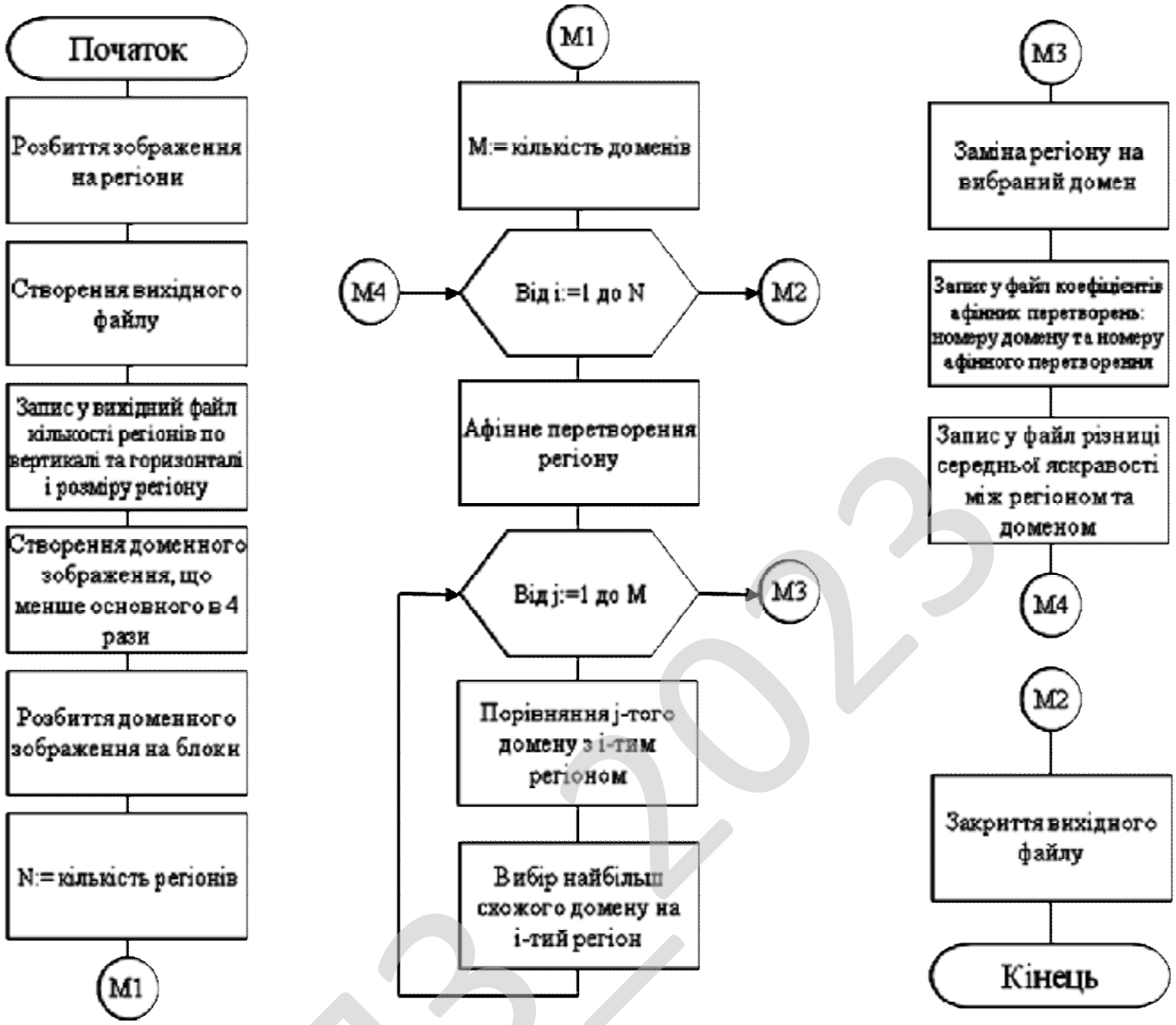

Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми стиснення зображення за допомогою фракталів

Якщо користувач обрав часткову декомпресію, тоді виконуються наступні дії:

– Вводиться кількість ітерацій.

– Запускається цикл по кількості ітерацій, у якому до зображення R_0 застосовується відображення W.

На цьому підпрограма завершує свою роботу.

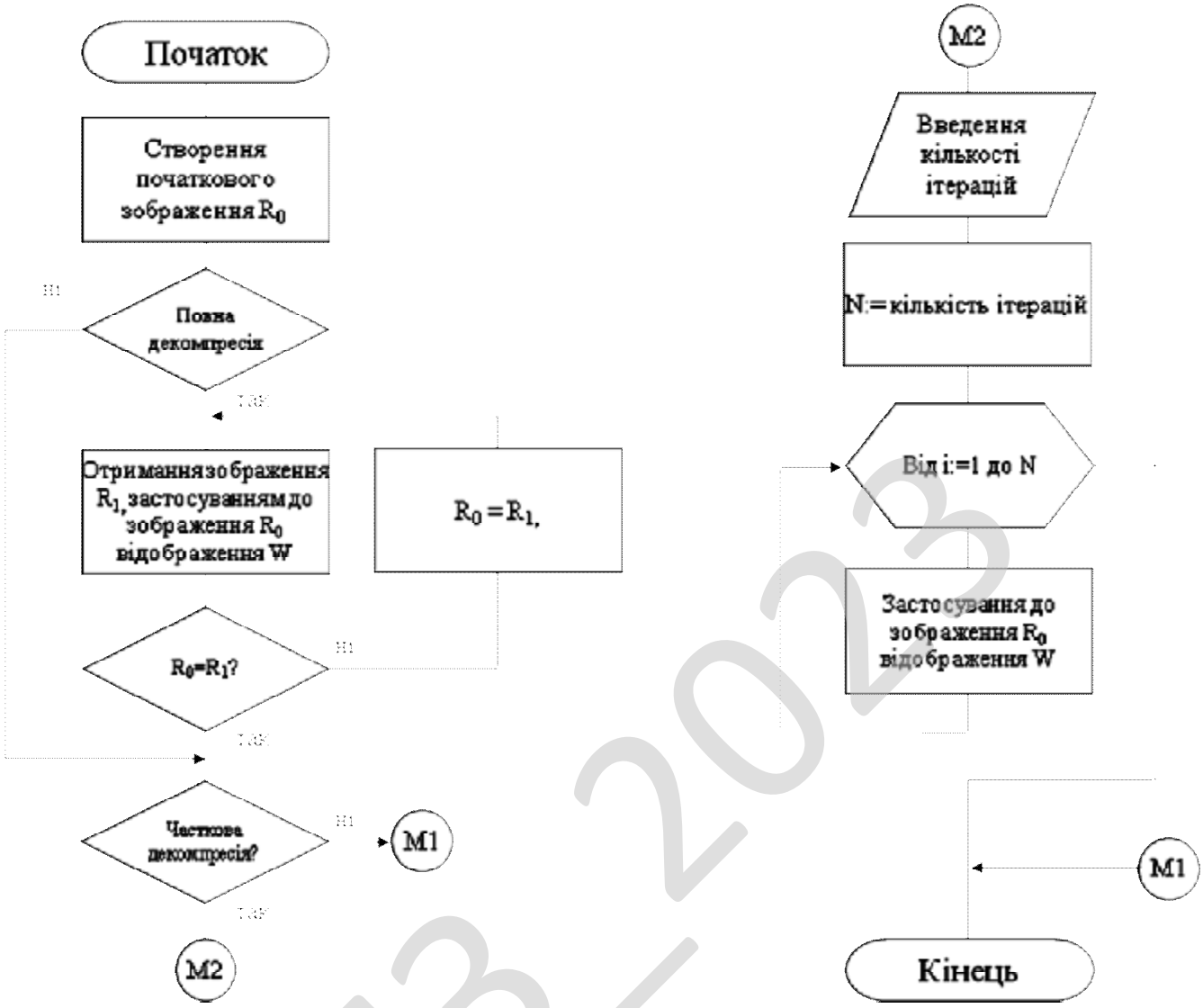

Рисунок 4.3 – Блок-схема алгоритму роботи підпрограми декомпресії зображення за допомогою фракталів

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;
```

Опишемо процедури, які реалізують алгоритм фрактального 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 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;
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму DES. DES є класичною мережею Фейштеля із двома гілками. Дані шифруються 64-бітними блоками, використовуючи 56-бітний ключ. Алгоритм перетворить за кілька раундів 64-бітний вхід в 64-бітний вихід. Довжина ключа дорівнює 56 бітам. Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти переставляються у відповідності зі стандартною таблицею. Наступний етап складається з 16 раундів однієї й тої ж функції, що використовує операції зрушення й підстановки. На третьому етапі ліва й права половини виходу останньої (16-й) ітерації міняються місцями. Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

Рисунок 4.4 – Загальна схема DES

Праворуч на рисунку показаний спосіб, яким використовується 56-бітний ключ. Спочатку ключ подається на вхід функції перестановки. Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зрушення й перестановки. Функція перестановки та сама для кожного раунду, але підключи K_i для кожного раунду виходять різні внаслідок повторюваного зрушення біт ключа.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс користувача головного вікна програми. З нього видно, що головне вікно складається з наступних блоків:

– Блок меню.

– Блок виведення початкового зображення та кінцевого результату.

– Блок виведення параметрів.

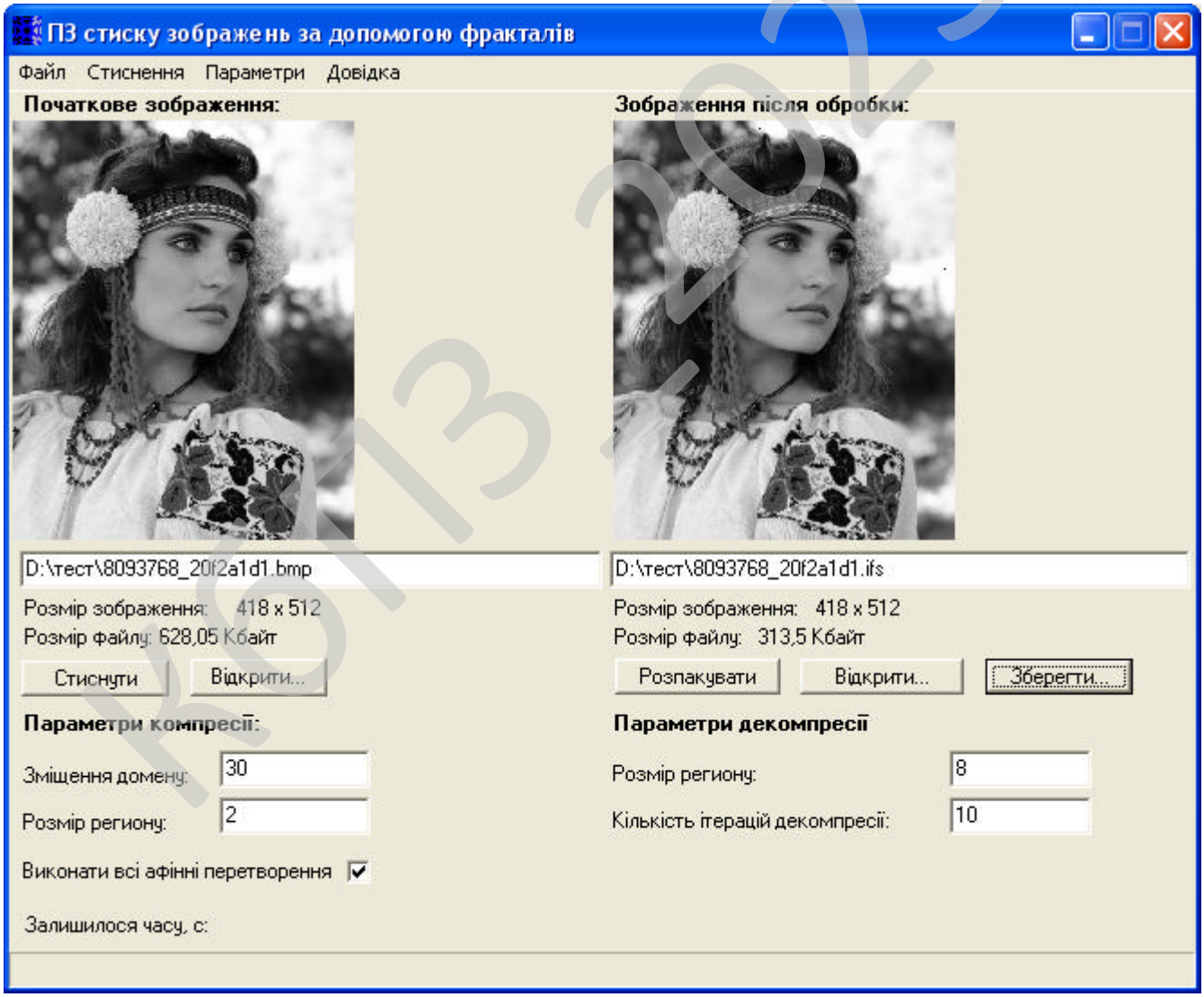

Рисунок 5.1 – Головне вікно програми – стиснення з розміром регіону рівним 2

Блок меню складається з наступних елементів:

– Файл.

– Стиснення.

– Параметри.

– Довідка.

Блок виведення початкового зображення та кінцевого результату складається з наступних елементів:

– Виведення початкового зображення.

– Виведення кінцевого зображення.

Початкове зображення наведено на рисунку 5.2.

Рисунок 5.2 – Вікно перегляду зображення – початкове зображення

Стиснуте зображення з розміром регіону рівним 2наведено на рисунку 5.3.

Блок виведення параметрів складається з наступних елементів:

– Вікна обрання шляху до файлу, який необхідно стиснути, або розпакувати.

– Кнопки швидкого доступу до елементів програми: відкрити, стиснути, розпакувати, зберегти.

– Параметри компресії: зміщення домену, розмір регіону.

– Параметри декомпресії: розмір регіону, кількість ітерацій декомпресії.

Рисунок 5.3 – Вікно перегляду зображення – стиснуте зображення з розміром регіону рівним 2

На рисунку 5.4 наведено результат стиснення з розміром регіону рівним 10. Як видно, відбувається дуже велика зернистість, й втрати частини зображення.

Більш детально зображення після обробки наведено на рисунку 5.5.

На рисунку 5.6 наведено вікно довідки, з якого можливо отримати наступні дані: тема проекту, виконавець проекту, керівник проекту, місце виконання проекту.

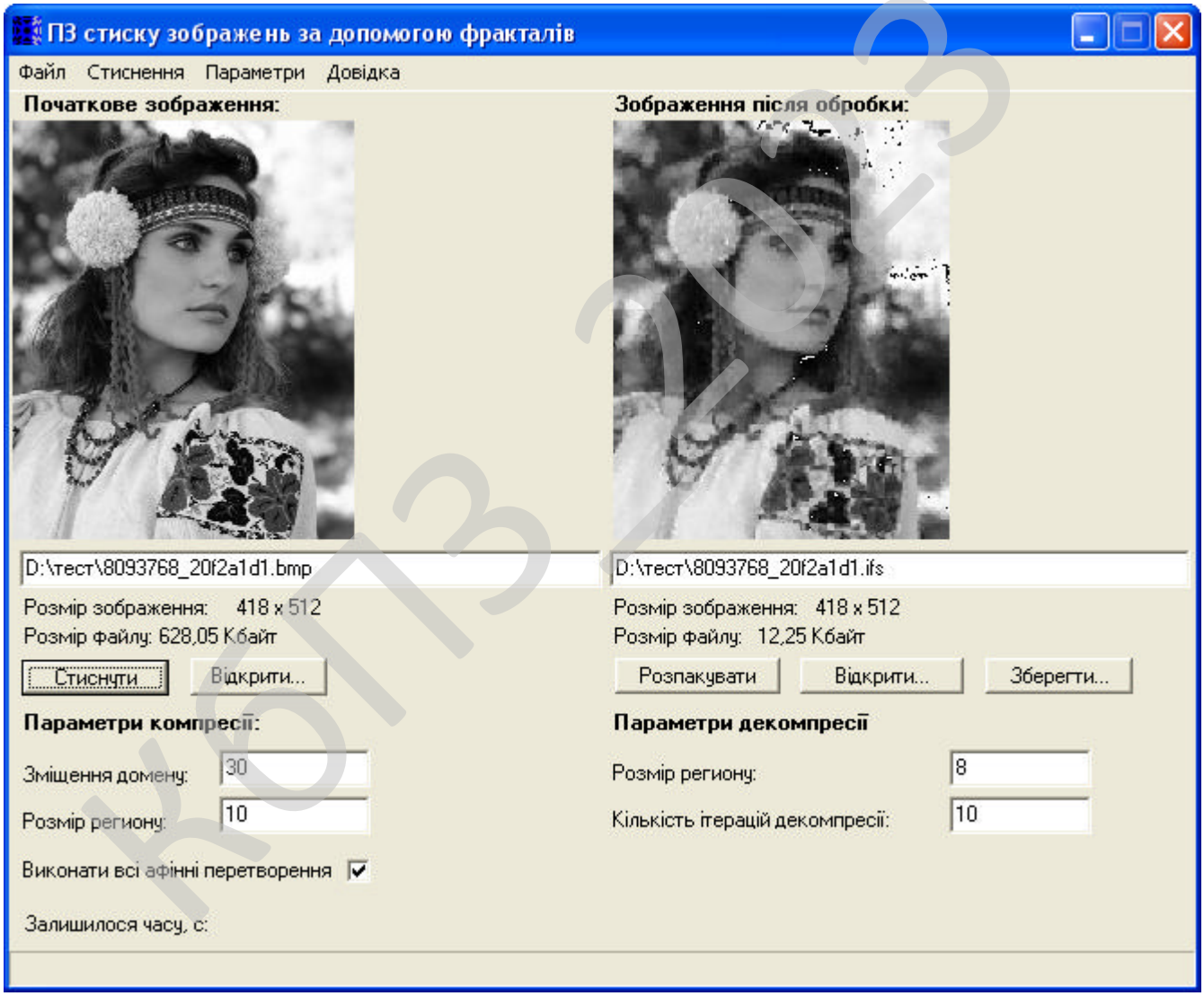

Рисунок 5.4 – Головне вікно програми – стиснення з розміром регіону рівним 10

Рисунок 5.5 – Вікно перегляду зображення – стиснуте зображення з розміром регіону рівним 10

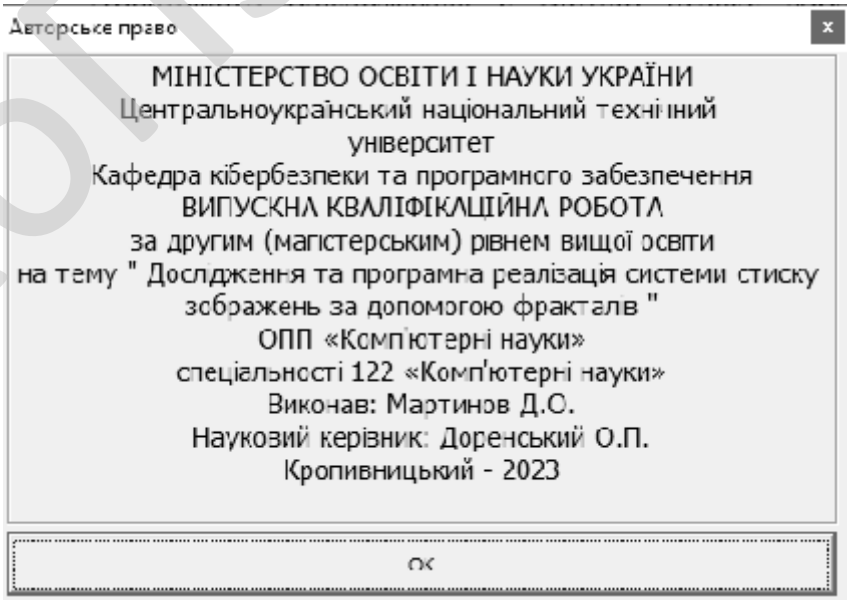

Рисунок 5.6 – Вікно довідки

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи стиску зображень за допомогою фракталів.

Метою розробки є дослідження та програмна реалізація системи стиску зображень за допомогою фракталів.

Об'єктом дослідження є процес стиску зображень за допомогою фракталів.

Предметом дослідження є методи стиску зображень за допомогою фракталів.

Методи дослідження базуються на методах обробки зображень, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод стиску зображень за допомогою фракталів.

– Розроблено вітчизняний продукт стиску зображень за допомогою фракталів, 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 24 днів (один місяць).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи стиску зображень за допомогою фракталів.

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	320
3. Запланований термін розробки, днів	Fpq	24 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	32000
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	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

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

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

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

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

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

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

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

$$T_{уточн} = 5{,}5 \cdot (1 \cdot 1{,}09 \cdot 1{,}30 \cdot 0{,}91 \cdot 1 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1 \cdot 0{,}87 \cdot 1{,}10 \cdot$$
$$\cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}10 \cdot 1 \cdot 1 \cdot 1{,}10) = 12{,}9 \text{ люд-міс.}$$

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

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

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

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

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

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 12{,}9^{0{,}33+0{,}2(1{,}027-1{,}01)} \cdot 50 = 94 \text{ люд/день.}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	94	Ф 7.1- 7.4
Впровадження	15	Д13
Всього	143	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{143 \cdot 1}{24\text{-}3} = 6,8 \text{ ставки.}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	10	900	15
Монітор	60	10	600	10
Клавіатура	30	10	300	5
Маніпулятор «мишка»	30	10	300	5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	300	750	12,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	56,16

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{56 \cdot 1}{1,2} = 47 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 47/(24 \cdot 8) = 0{,}25 \text{ ставки.}$$

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2012 R2, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	0,25	21000	5250
Продакт-менеджер	0,5	17136	8568
Інженер-програміст	6,8	19000	129200
Інженер-електронщик	0,25	16000	4000
Інженер-системотехнік	0,25	16000	4000
Адміністратор мережі	0,5	19000	9500
Системний програміст	0,25	16000	4000
Дизайнер WEB	0,5	16000	8000
Інженер-верстальник	0,25	18000	4500
Бухгалтер-економіст	0,5	15000	7500
Всього за період розробки	$R_{сп}$ = 10,05	-	$Ф_{роб}$ = 184518

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{184518}{10,05 \cdot 24} = 765 \; грн.$$

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:

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

де: $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 13 робочих місць;

$S_{у}$ – питома площа на одне робоче місце, $м^2$;

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

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 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:

$$Б_{уд} = 13 \cdot 8 \cdot 20000 = 2080000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 208000 грн.

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

$$I_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

де: $Ц_{м}$ – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 13 \cdot 3500 = 45500 \text{ грн.}$$

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 за пропозицією інтернет ресурсу hotline за 24.10.23 – джерело https://hotline.ua

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11457
Системний блок		7509
Процесор	Intel Core i3-8145U (2 (4) ядра по 2.1 - 3. GHz), 4 MB Smart Cache	-
Системна плата	Intel SoC platform, NUC8i3PN, 3x USB 3.1 2x USB 2.0, 1x USB Type-C, 2x Audio, 1 LAN (RJ-45), 1x HDMI, 1x DisplayPort	-
Відеокарта	Інтегрована Intel UHD Graphics 610 (до 179 МВ з ОЗП)	-
Жорсткий диск	SSD 240 Gb	-
Оперативна пам’ять	8 GB DDR4	-
DVD-привод	-	-
Корпус	Lenovo ThinkCentre M630e Tiny USFF, 65W	-
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	-
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V RUSSIA (BE525-RS)	1348

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	13	11457	14894,1	163835,1
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	297	3267
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	185653,6

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	2080000	-	-
2. Передавальні пристрої	208000	-	-
Всього по групі	2288000	5	114400
Група 4			
3. Обчислювальна техніка	185654	-	-
Всього по групі	185654	50	92827
Група 5, 6			
4. Вимірювальні пристрої	3999	25	1000
5. Транспортні засоби	97500	20	19500
6. Господарський інвентар	45500	25	11375
Всього по групі	146999	-	31875
7. Нематеріальні активи	32000	10	3200
Разом	$К_р$ = 2652653		$А_р$ = 242302

Примітка: вартість автомобіля взята по даним з прайс-листа автосалону автотрейдінг, вкладки автобазар, джерело http: //www.auto-trading.com.ua/sale/lot20772.html, складає 97500 грн.

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: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 765 \cdot 143/320 = 342 \text{ грн.}$$

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, %.

$$З_д = 342 \cdot 10 \cdot 0{,}01 = 34 \text{ грн.}$$

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

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

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

$$С_{оц} = 0{,}01 \cdot 22(342+34) = 83 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 342 \cdot 15 \cdot 0{,}01 = 51 \text{ грн.}$$

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

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

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картриджів, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,2 пачек паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 200 \cdot 0{,}2 = 40 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 100):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 32,15 грн./шт.

$$З_{М2} = 32{,}15 \cdot 100 = 3215 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

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

$$З_М = (40 + 3215 + 1702)/320 = 15 \text{ грн.}$$

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

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

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

$$О_п = 342 \cdot 15 \cdot 0{,}01 = 51 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 320 прим.):

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

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

$$A_м = 242302 \cdot 1/(320 \cdot 12) = 63 \text{ грн.}$$

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

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

$$C_п = 342+34+83+51+15+51+63 = 639 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 639 = 320 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	342
2. Додаткова зарплата виконавців	$З_д$	34
3. Відрахування на соціальні потреби	$С_{оц}$	83
4. Загальногосподарські витрати	$Г_{осп}$	51
5. Витрати на матеріали	$З_М$	15
6. Освоєння нових операційних систем, мов програмування	$О_п$	51

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	63
8. Повна собівартість програмного забезпечення	$С_п$	639
9. Плановий прибуток	$П_р$	320
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	959
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	191,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	1150,8

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	1151
Всього капітальних витрат	–	1151

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	$З_р$	25498	19800
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	576
Всього витрат за рік	$І$	25498	20376

Витрати на профілактичні роботи:

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

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 190 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 190 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 25498 \text{ грн},$$

до:

$$З_{р\,нов} = 150 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 19800 \text{ грн}.$$

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

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

$$З_{ел\ баз} = З_{ел\ нов}$$

Витрати 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	–	1151	–	575,5
Всього відрахувань	-	–	1151	–	575,5

7.8 Визначення економічної ефективності програмної продукції

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

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

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

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

$$Ев = (959 - 639) \cdot 320 - (0,05 \cdot 2288000 + 0,5 \cdot 185654 +$$
$$+0,25 \cdot 49499 + 0,2 \cdot 97500 + 0,1 \cdot 32000) \cdot 1/12 = 82208\ грн.$$

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

$$T_в = \frac{364653}{(959-639)\cdot 320\cdot 12/1} = 0{,}3 \ року .$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	320
2. Повна собівартість розробленої програми	Грн.	639
3. Ціна розробленої програми	Грн.	959
4. Плановий прибуток від реалізації розробленої програми	Грн.	320
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2652653
7. Загальний прибуток від реалізації програмної продукції	Грн.	102400
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	82208
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	0,3
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1151
11.Величина економічного ефекту у користувача програмної продукції	Грн.	4547
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,22

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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.

$$E_{сп} = (25498\text{-}20376)\text{-}0{,}5 \cdot 1151 = 4547 \text{ грн.}$$

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

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

$$T_{сп} = \frac{1151}{(25498\text{-}20376)} = 0{,}22 \text{ року}.$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Загальна комп'ютеризація суспільства призвела до того, що використання комп'ютерів стало повсюдним у всіх сферах економіки та народного господарства. Застосування персональних комп'ютерів і ЕОМ дозволило значно підвищити продуктивність праці, змінити характер і зміст праці [4]. Комп'ютеризація, поряд з незаперечними перевагами, тягне за собою і багато проблем. Для того, щоб активне застосування комп'ютерних технологій не стало додатковим чинником погіршення здоров'я, вкрай необхідно щоб робоче місце відповідало гігієнічним вимогам. Темою дипломного проекту є розробка та дослідження та реалізація програмного продукту, тому актуально буде розгляд умов праці програміста.

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

В кожній ІТ компанії є трудові відносини з працівниками. Згідно закону України "Про охорону праці" [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні.

– Положення про охорону праці.

– Накази з охорони праці.

– Журнали реєстрації та інструктажу.

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

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

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

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

8.2 Аналіз умов праці на робочому місці програміста

Робота програміста пов'язана з постійною роботою на ЕОМ, яка відбувається у кімнаті розмірами 5м×7,2м×2,8м. Одна з її більших стін має шість двостулкових вікон, розмірами 2,2м×1,8м, які виходять на північний схід. Вікна розташовані рівномірно по всій довжині стіни. Підлога в кімнаті покрита леноліумом, всі стіни пофарбовані світло оранжевого кольору до висоти 2,8м, 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 1250 мм×850 мм. Висота столів 750 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, 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 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 рівні звукового тиску в робочому приміщенні не повинні перевищувати в октавних смугах із середньо геометричними частотами наступних значень, наведених у таблиці 8.1.

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 8.1 – 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

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 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

8.3 Розробка заходів з умов поліпшення охорони праці

Аналіз умов праці на робочому місці інженера-програміста показав, що на робочому місці не виконуються вимоги ергономіки. Для виконання їх можна запропонувати заміну не регульованого сидіння на крісло з регульованими ергономічними параметрами, а також заміну використовуваного столу на робоче місце оператора ЕОМ.

Різними фірмами в сукупності розроблено понад 11 схем регулювань параметрів робочого крісла, які забезпечують: плавне переміщення сидіння по

висоті за допомогою газової пружини; плавна зміна нахилу спинки і сидіння; регулювання пружинного противотиску спинки крісла на спину оператора; перестановку спинки по висоті; зміна глибини сидіння шляхом зміни вигину краю сидіння; синхронне повторення рухів оператора сидінням і спинкою в правильному кутовому співвідношенні; синхронне повторення спинкою крісла рухів верхньої частини тулуба того, що сидить; амортизацію сидіння.

Найбільш популярними моделями комп'ютерних крісел, які володіють чотирма основними регулюваннями (висоти сидіння, висоти, глибини і нахилу спинки), є італійські, фінські моделі «Senior», «Volos», «Ergo», «Toronto», «Bini», «Metro», «NewStar», «Capris», «Fenix», «Xenus», «Quintus» і т.д. Подібні крісла, відрегульовані відповідно до зростання і ваги оператора, а також характеру виконуваної роботи, дозволяють понизити навантаження на опорно-руховий апарат людини, що працює за комп'ютером.

8.4 Розрахункова частина

Проводемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 5 м, довжина – 7,2 м, висота – 2,8 м.

У зазначеному приміщенні працює 5 людей.

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

$$F=ESKZ/n,$$

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

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

S – площа освітлюваного приміщення (у нашому випадку S=5 х 7,2 = 36 $м^2$);

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

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

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

Обчислимо індекс приміщення за формулою:

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

де: *S* – площа приміщення, *S* = 36 *м*^{2}*;*

h – розрахункова висота підвісу, *h = 3 м (співпадає з в*исотою стелі, т.я. лампи освітлення закріплюються на стелі)*;*

A – ширина приміщення, *A = 5 м;*

B – довжина приміщення, *B =* 7,2 *м.*

Підставимо всі значення у формулу та визначемо індекса приміщення: i=1,4.

Знаючи індекс приміщення, за знаходимо *n = 0,29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам).* Підставимо всі значення у формулу, визначемо світловий потік: *F=77478 Лм.*

Для розрахунку дудемо використовувати *світлодіодні панелі LED панель 42Вт 6000К SUNLED 000000127*, світловий потік яких $F_л$ *= 3990 Лм.*

Число ламп визначається по формулі:

$$N=F/F_л$$

де: F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекса приміщення: $N=$

$77478/ 3990=19,4$ *шт.*

Приймаємо необхідну кількість *світлодіодних світильників* 20 шт.

8.5 Висновки до розділу

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

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з умов поліпшення охорони праці.

9 ОСНОВНІ ВИСНОВКИ

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

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 стиску зображень за допомогою фракталів.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем стиску зображень за допомогою фракталів.

– Досліджена система стиску зображень за допомогою фракталів.

– На основі отриманих результатів досліджень створена програмна реалізація системи стиску зображень за допомогою фракталів.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання стиску зображень за допомогою фракталів.

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 RAD Studio 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 DES.

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 4547 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,22 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мартинов Д.О. Дослідження та програмна реалізація системи стиску зображень за допомогою фракталів // 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. G.E. Qien, Z. Baharav, S. Lepsqy, E. Karnin. A new improved collage theorem with appplications to multiresolution fractal image coding. In Proc. ICASSP, 1994.

5. Saupe, D., Hamzaoui, R., Hartenstein, H., Fractal image compression – An introductory overview, in: Fractal Models for Image Synthesis, Compression and Analysis, D. Saupe, J. Hart (eds.), SIGGRAPH'96 Course Notes, ACM, New Orleans, Louisiana, Aug. 1996.

6. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms. The MIT Press. 2022 1677 p.

7. Will Grant. 101 UX Principles. Packt Publishing. 2022. 432 p.

8. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

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

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

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

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

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

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

15. 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.

16. 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.

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

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

19. 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.

20. 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,

21. 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.

22. 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

23. 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.

24. 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.

25. 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.

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

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи стиску зображень за допомогою фракталів.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи стиску зображень за допомогою фракталів.

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 стиску зображень за допомогою фракталів;

– 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 RAD Studio 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 повинен бути розглянутий аналіз умов праці на робочому місці програміста.

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 – 118 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 11.12.2023 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

_____ Доренський О.П.

Дослідження та програмна реалізація системи стиску зображень за допомогою фракталів

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2023 року

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.
```

FractalCompression.pas - файл алгориму фрактального стиснення

```
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 кольороякравість
    Ifs: TIfsRec; // Параметри, що обчислюються при компресії
  end;

  TDomainRec = packed record
    MeanColor: Integer; // Усереднена кольороякравість
  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.
```

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.
```