РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: реалізація пошуку прихованої інформації в растрових зображеннях.

Мета кваліфікаційної роботи: розробка програмного забезпечення для виявлення прихованої інформації в растрових зображеннях на основі досліджених властивостей стеганографічних контейнерів та розроблених алгоритмів їх пошуку та створення системи для виявлення наявності стеганографічних даних.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано платформу для розробки, виконано проєктування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні додатка, що надає можливість здійснювати аналіз статистичних розподілів бітів у зображеннях, що містять стеганографічні дані, виявляти їх характерні риси, та за допомогою програми виявляти їх в файлах з растровими зображеннями.

Актуальність інформаційної системи визначається тим, що стеганографія може використовуватися зловмисниками для безпечної передачі секретних даних, які в результаті можуть виявитися небезпечними для суспільства, а найпростіше передавати такі дані через зображення.

Список ключових слів: СТЕГАНОГРАФІЯ, РАСТРОВЕ ЗОБРАЖЕННЯ, ШИФРУВАННЯ, КОМП’ЮТЕР, АЛГОРИТМ, ПРОЕКТУВАННЯ, ФАЙЛ, ДОДАТОК.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: implementation of the search for hidden information in raster images.

The purpose of the qualification work: development of software for detecting hidden information in raster images based on the studied properties of steganographic containers and developed algorithms for their search and creating a system for detecting the presence of steganographic data.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes existing solutions, selects a platform for development, designs and develops the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download of the program, describes the program.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical value is to create an application that allows you to analyze the statistical distributions of bits in images containing steganographic data, to identify their characteristics, and with the help of the program to detect them in files with raster images.

The relevance of the information system is determined by the fact that steganography can be used by attackers to securely transmit classified data, which in turn can be dangerous to society, and the easiest way to transmit such data through images.

List of keywords: STEGANOGRAPHY, PRAGUE IMAGE, ENCRYPTION, COMPUTER, ALGORITHM, DESIGN, FILE.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ДКП – дискретне косинусне перетворення;

ІС – інформаційна система;

НЗБ – найменш значущі біти;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

ЦВЗ – цифровий водяний знак;

IDE – Integrated development environment, інтегроване середовище

LSB – Least Significant Bit, найменший значущий біт;

RGB – Red, Green, Blue палітра червоний, зелений, синій.

ЗМІСТ

ВСТУП

Проблема приховування даних хвилює людство з давніх часів. XXI століття – епоха інформатики та інформатизації, основною цінністю є інформація. Технологія дає можливість передавати і зберігати все більші обсяги інформації. Це благо має й зворотний бік. Інформація стає значно вразливішою з різних причин:

— зростання обсягів збережених і переданих даних;

— розширення кола користувачів, що мають доступ до інформаційних систем;

— ускладнення інформаційних систем.

Рано чи пізно перед кожною інформаційною системою постає важливе питання забезпечення захисту даних шляхом приховування інформації від осіб, для яких вона не призначена. Виконати це завдання в рамках самої системи можна за рахунок впровадження системи аутентифікації, завдяки чому доступ зможуть отримати виключно авторизовані користувачі. Однак якщо зловмисникові вдасться отримати фізичний доступ до даних, то аутентифікація стане марною. Буде порушена конфіденційність даних (доступ не тільки тим, кому призначені), вони можуть бути змінені, що також ще і порушить їх цілісність. Для запобігання подібного результату застосовується шифрування даних.

В наш час комп’ютер відіграє роль «великої інформаційної скриньки». Суспільство накопичує різноманітні дані та обробляє їх. З набуттям цінності інформація потребує шифрування та приховування. Для цього існує стеганографія.

Слово «стеганографія» з грец. значить «тайнопис».

Тайнопис - це методи кодування, особливістю яких є обов'язкове збереження в таємниці криптографічного алгоритму від третьої сторони, що певним чином обмежує їх функціональні можливості

Історично цей напрямок було створено першим, проте потім він був витіснений криптографією. Тайнопис здійснюють різними способами. Рисою, що об’єднує ці способи, є повідомлення, яке потрібно приховати. Його вбудовують в якийсь сторонній об’єкт, який не привертає уваги й відправляють адресату. У криптографії присутність у посланні шифрованого повідомлення само по собі може привернути увагу зловмисників, при стеганографії прихований зв’язок залишається непомітним. Але в той же час і стеганографія може використовуватися зловмисниками для безпечної передачі секретних даних, які в результаті можуть виявитися небезпечними для суспільства. Виявляється, що найпростіше передавати такі дані через зображення.

Актуальність роботи стає зрозумілою, якщо розглянути практичні задачі, для яких стеганографія актуальна:

- непомітна передача інформації;
- приховане зберігання інформації;
- недекларованe зберігання інформації;
- захист виключного права;
- захист авторського права;
- захист автентичності документів;
- індивідуальний відбиток;
- водяний знак у DPL системах;
- прихована передача керованого сигналу;
- підтвердження достовірності переданої інформації;
- невідчужуваність інформації;
- стеганографічне відстеження.

Тому, створення алгоритмів виявлення наявності стеганографічних даних в файлах є актуальним.

Метою кваліфікаційної роботи є розробка програмного забезпечення для виявлення прихованої інформації в растрових зображеннях на основі досліджених властивостей стеганографічних контейнерів та розроблених

алгоритмів їх пошуку та створення системи для виявлення наявності стеганографічних даних.

Об’єктом розробки є програма реалізації пошуку прихованої інформації в растрових зображеннях.

В ході виконання кваліфікаційної роботи виконується аналіз сучасних стеганографічних алгоритмів, порівнюються методи виявлення стеганографічних даних у растрових зображеннях, здійснюється аналіз статистичних розподілів бітів у зображеннях, що містять стеганографічні дані, виявляються їх характерні риси, що можуть бути використані для побудови алгоритму, проектується та програмується алгоритм виявлення стеганографічних даних.

Практичне значення полягає у створенні додатка, що надає можливість здійснювати аналіз статистичних розподілів бітів у зображеннях, що містять стеганографічні дані, виявляти їх характерні риси, та за допомогою програми виявляти їх в файлах з растровими зображеннями.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Основні поняття інформаційної безпеки

Тема захисту інформації є актуальною в усьому світі, у різних сферах життя, оскільки сучасне суспільство цілком залежить від комп'ютерних систем. Українці не є виключенням, тим більше, проблема захисту та безпечної передачі даних при проведенні військової операції на Сході України є досить актуальною. У XXI столітті, в епоху інформаційних війн, перемогу здобуває той, хто володіє достовірною та вчасно отриманою інформацією.

Сучасний світ – світ високих технологій та створених на їх основі винаходів. Інформація в цифровій формі в 21 столітті – це те, без чого наш світ припинив би своє існування в тому вигляді, в якому ми його бачимо, вже через декілька днів. Інформація є найціннішим ресурсом для людини в сучасному суспільстві, і, як за будь-який цінний ресурс, за неї ведеться справжня війна. Тому інформація повинна бути надійно захищена. Як сотні років тому людину захищав меч та щит, пізніше – бронежилет, так і захист цифрових відомостей заснований на аналогічних принципах. І одним з найнадійніших технологій захисту є шифрування інформації.

Шифрування - перетворення інформації у формат, складний для сприйняття для того, щоб приховати її від сторонніх осіб, та в той же час, для надання авторизованим користувачам доступу до неї. В інформації не має сенсу, якщо її не можна сприйняти правильним чином. Інформація, яку може сприйняти тільки авторизована на те особа, не є корисною для всіх інших, а, відповідно, не має цінності та не варта витрат ресурсів для отримання доступу до неї.

Тайнопис − засекречене письмо або текст, ним писаний.

Найперші форми тайнопису вимагали не більше ніж аналогів олівця та паперу, оскільки в ті часи більшість людей не могли читати. Поширення писемності викликало потребу саме в криптографії.

Криптографія (від грецького κρυπτός − прихований і gráphein − писати) − наука про математичні методи забезпечення конфіденційності, цілісності і автентичності інформації. Розвинулась з практичної потреби передавати важливі відомості найнадійнішим чином. Для математичного аналізу криптографія використовує інструментарій абстрактної алгебри та теорії ймовірностей.

Тривалий час під криптографією розумілось лише шифрування − процес перетворення звичайної інформації (відкритого тексту) в незрозуміле «сміття» (тобто, шифротекст). Дешифрування − зворотний процес відтворення інформації із шифротексту. Шифром називається пара алгоритмів шифрування/розшифрування. Дія шифру керується як алгоритмами, та, в кожному випадку, ключем. Ключ − секретний параметр (в ідеалі, відомий лише двом сторонам) для окремого контексту під час передачі повідомлення. Ключі мають велику важливість, оскільки без змінних ключів алгоритми шифрування легко зламуються і непридатні для використання в більшості випадків. Історично склалось так, що шифри часто використовуються для шифрування та дешифрування, без виконання додаткових процедур, таких як аутентифікація або перевірка цілісності.

Основними типами класичних шифрів є перестановочні шифри, які змінюють порядок літер в повідомленні, та підстановочні шифри, які систематично замінюють літери або групи літер іншими літерами або групами літер. Прості варіанти обох типів пропонували слабкий захист від досвідчених супротивників.

Шляхом застосування шифрування намагаються зберегти зміст спілкування в таємниці, подібно до шпигунів, військових лідерів та дипломатів. Збереглися також відомості про деякі з ранніх єврейських шифрів.

1.1.2. Основні поняття стеганографії

Стеганографія – тайнопис, при якому повідомлення, закодоване таким чином, що не виглядає як повідомлення – на відміну від криптографії. Непосвячена людина принципово не може розшифрувати повідомлення, бо не знає та навіть не здогадується про факт самого його існування. На відміну від криптографії, яка приховує зміст повідомлення.

Якщо криптографія приховує зміст повідомлення, то стеганографія приховує сам факт існування повідомлення.

Розглядаючи програмні засоби захисту, доцільно спинитись на стеганографічних методах. Слово «стеганографія» означає приховане письмо, яке не дає можливості сторонній особі дізнатися про його існування. Одна з перших згадок про застосування тайнопису датується V століттям до н. е. Сучасним прикладом є випадок роздрукування на ЕОМ контрактів з малопомітними викривленнями обрисів окремих символів тексту - так вносилась шифрована інформація про умови складання контракту.

Комп'ютерна стеганографія базується на двох принципах. По-перше, аудіо- і відеофайли, а також файли з цифрованими зображеннями можна деякою мірою змінити без втрати функціональності. По-друге, можливості людини розрізняти дрібні зміни кольору або звуку обмежені. Методи стеганографії дають можливість замінити несуттєві частки даних на конфіденційну інформацію. Сімейна цифрова фотографія може містити комерційну інформацію, а файл із записом сонати Гайдна - приватний лист.

Але найчастіше стеганографія використовується для створення цифрових водяних знаків. На відміну від звичайних їх можна нанести і відшукати тільки за допомогою спеціального програмного забезпечення - цифрові водяні знаки записуються як псевдовипадкові послідовності шумових сигналів, згенерованих на основі секретних ключів. Такі знаки можуть забезпечити автентичність або недоторканість документа, ідентифікувати автора або власника, перевірити права дистриб'ютора або користувача, навіть якщо файл був оброблений або

спотворений.

Щодо впровадження засобів програмно-технічного захисту в ІС, розрізняють два основні його способи:

— додатковий захист - засоби захисту є доповненням до основних програмних і апаратних засобів комп'ютерної системи;

— вбудований захист - механізми захисту реалізуються у вигляді окремих компонентів ІС або розподілені за іншими компонентами системи.

Перший спосіб є гнучкішим, його механізми можна додавати і вилучати за потребою, але під час його реалізації можуть постати проблеми забезпечення сумісності засобів захисту між собою та з програмно-технічним комплексом ІС. Вмонтований захист вважається більш надійним і оптимальним, але є жорстким, оскільки в нього важко внести зміни. Таким доповненням характеристик способів захисту зумовлюється те, що в реальній системі їх комбінують.

Наприкінці 90-х років виділилося кілька напрямків стеганографії:

– класична стеганографія;

– комп'ютерна стеганографія;

– цифрова стеганографія;

– мережева стеганографія.

Класична - використання симпатичних (невидимих) чорнил. Процес запису здійснюється наступним чином: перший шар – наноситься важливий запис невидимим чорнилом, другий шар – запис видимими чорнилом, що нічого не значить.

Текст, записаний такими чорнилом, проявляється лише за певних умов (нагрівання, освітлення, хімічний проявник).

Винайдені ще в I столітті н. е. Філоном Александрійським, ними користувались як в середньовіччі, так і в новітній час, наприклад, у листах революціонерів з російських в'язниць. Написаний звичайним молоком текст на

папері між рядків видимого тексту, проявляється при нагріванні над полум'ям (зазвичай свічки).

Існує також чорнило з хімічно нестабільним пігментом. Написане цими чорнилами виглядає як написане звичайною ручкою, але через певний час нестабільний пігмент розкладається, і від тексту не залишається і сліду. Хоча при використанні звичайної кулькової ручки текст можливо відновити по деформації паперу, цей недолік можна усунути за допомогою м'якого вузла, на зразок фломастера.

Симпатичними чорнилами можуть слугувати найрізноманітніші речовини: лимонна кислота, віск, яблучний сік, молоко, сік цибулі, слина, пральний порошок, аспірин, крохмаль з різними хімічними чи фізичними «декодерами»: температура, сода, йод, солі, заліза, ультрафіолетове світло, для воску навіть крейда чи зубний порошок.

Комп'ютерна стеганографія – напрям класичної стеганографії, заснований на особливостях комп'ютерної платформи. Приклади – стеганографічна файлова система StegFS для Linux, приховування даних в невикористовуваних областях форматів файлів, підміна символів в назвах файлів, текстова стеганографія тощо.

Цифрова стеганографія – напрям класичної стеганографії, заснований на приховуванні або впровадженні додаткової інформації в цифрові об'єкти, викликаючи при цьому деякі спотворення цих об'єктів. Але, як правило, дані об'єкти є мультимедіа-об'єктами (зображення, відео, аудіо, текстури 3D-об'єктів) та внесення спотворень, які знаходяться нижче межі чутливості середньостатистичної людини, не призводить до помітних змін цих об'єктів.

Мережева стеганографія - прихована інформація передається через комп'ютерні мережі з використанням особливостей роботи протоколів передачі даних. Цей термін вперше ввів Кжиштоф Щипьорский (Krzysztof Szczypiorski) в 2003 році. Типові методи мережевої стеганографії включають зміну властивостей одного з мережевих протоколів та можливе використання

взаємозв'язку між двома або більше різними протоколами для більш надійного приховування факту передачі секретного повідомлення.

Крім того, в цифрованих об'єктах, тобто таких, що спочатку мають аналогову природу, завжди присутній шум квантування; також, при відтворенні цих об'єктів з'являється додатковий аналоговий шум і нелінійні спотворення апаратури, все це сприяє більшій непомітності прихованої інформації.

Існуючі алгоритми вбудовування таємної інформації можна поділити на декілька підгруп:

— працюючі з самим цифровим сигналом. Наприклад, метод LSB (Least Significant Bit);

— «впаювання» прихованої інформації. В даному випадку відбувається накладення приховуваного зображення (звуку, іноді тексту) поверх оригіналу. Часто використовується для вбудовування ЦВЗ (цифровий водяний знак).

— використання особливостей форматів файлів. Сюди можна віднести запис інформації в метадані або в різні інші не використовувані зарезервовані поля файлу.

Найчастіше стеганографія використовується для створення цифрових водяних знаків. На відміну від звичайних їх можна нанести і відшукати тільки за допомогою спеціального програмного забезпечення - цифрові водяні знаки записуються як псевдовипадкові послідовності шумових сигналів, згенерованих на основі секретних ключів. Такі знаки можуть забезпечити автентичність або недоторканість документа, ідентифікувати автора або власника, перевірити права дистриб'ютора або користувача, навіть якщо файл був оброблений або спотворений.

Цифровий водяний знак (ЦВЗ) - технологія, створена для захисту авторських прав мультимедійних файлів та інтелектуальної власності контейнера (Intellectual Property). Зазвичай цифрові водяні знаки невидимі. Однак ЦВЗ можуть бути видимими на зображенні або відео. Зазвичай це інформація являє собою текст або логотип, який ідентифікує автора.

Стеганографія застосовує ЦВЗ, коли сторони обмінюються секретними

повідомленнями, впровадженими в цифровий сигнал. Використовується як засіб захисту документів з фотографіями - паспортів, водійських посвідчень, кредитних карт з фотографіями.

Стегоаналіз - розділ стеганографії; наука про виявлення факту передачі закритою інформацією в аналізованому повідомленні. У деяких випадках під стеганалізмом розуміють також витягування прихованої інформації з вмісту її повідомлення і (якщо це необхідно) подальшу її дешифровку.

1.1.3. Характеристика растрових зображень

Растрова графіка (англ. raster graphics) є частиною комп'ютерної графіки, яка має справу зі створенням, обробкою та зберіганням растрових зображень. Растрове зображення є масивом кольорових точок (пікселів). Обробка растрової графіки здійснюється растровими графічними редакторами. Растрові зображення зберігаються у різних графічних форматах.

Растрове зображення - зображення, яке являє собою сітку (растр), зазвичай прямокутну, пікселів відображених на моніторі, папері та інших відображальних пристроях і матеріалах.

Характеристиками растрового зображення є:

— кількість пікселів - зазвичай вказують кількість пікселів по ширині і висоті (наприклад, 1024 × 768, 1920 × 1080);

— кількість використовуваних кольорів або глибина кольору (обсяг пам'яті в бітах, що використовуються для одного пікселя);

— колірний простір - RGB, CMYK, XYZ, YCbCr та ін;

— роздільна здатність - довідкова величина, яка вказує на рекомендований розмір зображення.

Растрові зображення редагують за допомогою растрових графічних редакторів. Створюються растрові зображення фотоапаратами, сканерами, безпосередньо в растровому редакторі, також шляхом експорту (растерізацією) з векторного редактора або у вигляді знімків екрану.

Растрові зображення зазвичай зберігаються в стислому вигляді. Залежно від типу стиснення може бути можливо або неможливо відновити зображення в точності таким, яким воно було до стиснення (стиснення без втрат або стиснення з втратами відповідно). Так само в графічному файлі може зберігатися додаткова інформація: про автора файлу, фотокамері і її налаштуваннях, кількості точок на дюйм при друку та ін.

Використовує алгоритми стиснення, засновані на зменшенні надмірності інформації.

— BMP або Windows Bitmap - зазвичай використовується без стиснення, хоча можливо використання алгоритму RLE;

— GIF (Graphics Interchange Format) - формат, який витісняється PNG та підтримує не більше 256 кольорів одночасно. Усе ще популярний через підтримку анімації, яка відсутня в чистому PNG, хоча програмне забезпечення дедалі більше підтримує формат APNG;

— PCX - застарілий формат, що дозволяв добре стискати прості зображення;

— PNG (Portable Network Graphics);

— TIFF підтримує великий діапазон зміни глибини кольору, різні колірні простору, різні настройки стиснення (як з втратами, так і без) і ін;

— RAW зберігає інформацію, безпосередньо одержувану з матриці цифрового фотоапарата або аналогічного пристрою без застосування до неї будь-яких перетворень, а також зберігає інформацію про налаштування фотокамери.

1.1.4. Огляд наявних аналогів

Для більшого розуміння предметної галузі кваліфікаційної роботи та розробки постановки завдання, виконаний аналіз наявних додатків, що забезпечують подібний функціонал:

1. Сайт hid.im. Дозволяє користувачам приховувати файли .torrent всередині зображень PNG.

2. DarkCryptTC і Проект «Зоря». Цю програму, можна назвати найбільш потужним стеганографічним рішенням. Вона підтримує більше сотні різних симетричних і асиметричних криптоалгоритмів. Включає в себе підтримку власної системи плагінів, призначеної для блокових шифрів (BlockAPI), текстову, аудіо та графічну стеганографію (включаючи реальну стеганографію JPEG), потужний генератор паролів і систему знищення інформації і ключів.

3. FFEncode. Цікава програма, яка приховує дані в текстовому файлі. Програма запускається з відповідними параметрами з командного рядка.

4. Steganos LockNote 1.0.5. Невелика і дуже зручна програма, що дозволяє зашифрувати будь-яку конфіденційну інформацію, збережену в невеликих замітках. Це можуть бути паролі, серійні номери або будь-які інші невеликі за обсягом дані. Будь-яка з зашифрованих нотаток після збереження стає звичайним виконуваним файлом (* .exe), відкрити який для читання або редагування можна тільки після введення пароля, заданого перед збереженням. Високу стійкість до злому забезпечує шифрування з 256-бітовим ключем AES, а зашифровані файли займають місця лише трохи більше, ніж оригінальні замітки.

1.2. Призначення розробки та область застосування

Стеганографія може використовуватися зловмисниками для безпечної передачі секретних даних, які в результаті можуть виявитися небезпечними для суспільства, а найпростіше передавати такі дані через зображення.

Методи стеганографії дають можливість замінити несуттєві частки даних на конфіденційну інформацію. Сімейна цифрова фотографія може містити комерційну інформацію, а файл із записом сонати Гайдна - приватний лист.

Але найчастіше стеганографія використовується для створення цифрових водяних знаків. На відміну від звичайних їх можна нанести і відшукати тільки

за допомогою спеціального програмного забезпечення - цифрові водяні знаки записуються як псевдовипадкові послідовності шумових сигналів, згенерованих на основі секретних ключів. Такі знаки можуть забезпечити автентичність або недоторканість документа, ідентифікувати автора або власника, перевірити права дистриб'ютора або користувача, навіть якщо файл був оброблений або спотворений.

Стеганографія застосовує ЦВЗ, коли сторони обмінюються секретними повідомленнями, впровадженими в цифровий сигнал. Використовується як засіб захисту документів з фотографіями - паспортів, водійських посвідчень, кредитних карт з фотографіями.

ЦВЗ можна також використовувати для виявлення потенційних піратів: під час продажу в зображення вбудовують інформацію про час продажу та інформацію про покупця. Ключовою відмінністю ЦВЗ від звичайного приховання інформації є наявність активного противника. Наприклад, використовуючи ЦВЗ для захисту авторського права, активний противник намагатиметься видалити чи змінити вбудовані ЦВЗ. Тому основною вимогою є стійкість вбудованих даних до атак. Таємність не є настільки важливою, як у прихованій комунікації.

Головна цінність розробленої системи полягає в тому, що створене в її рамках програмне забезпечення може бути використане для виявлення потенційних загроз різної природи. Запропонований алгоритм може бути також адаптований для інших видів стеганографічних контейнерів.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка програмного забезпечення для виявлення прихованої інформації в растрових зображеннях на основі досліджених властивостей стеганографічних контейнерів» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Метою кваліфікаційної роботи є розробка програмного забезпечення для виявлення прихованої інформації в растрових зображеннях на основі досліджених властивостей стеганографічних контейнерів та розроблених алгоритмів їх пошуку та створення системи для виявлення наявності стеганографічних даних.

Для виконання кваліфікаційної роботи необхідно:

– виконати аналіз сучасних стеганографічних алгоритмів;

– порівняти методи виявлення стеганографічних даних у растрових зображеннях;

– здійснити аналіз статистичних розподілів бітів у зображеннях, що містять стеганографічні дані;

– виявити їх характерні риси, що можуть бути використані для побудови алгоритму;

– спроектувати та запрограмувати алгоритм виявлення стеганографічних даних;

– здійснити тестування й відлагодження програми.

Розроблене програмне забезпечення повинно надавати можливість здійснювати аналіз статистичних розподілів бітів у зображеннях, що містять стеганографічні дані, виявляти їх характерні риси, та виявляти їх в файлах з растровими зображеннями.

Розроблений програмний додаток має виконувати наступні функції:

1. Виявляти приховане повідомлення в растрових 24-х бітних зображеннях BMP-формату.

2. Виявляти ймовірність наявності стегоданих в діапазоні від 0 до 100%, з найбільш точним результатом.

Створений додаток має бути протестований на конкретних прикладах та з різноманітними даними, що засвідчуватиме про його працеспроможність.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Програмна реалізація має складатися з чотирьох основних частин:

1. Зчитування файлу (зображення).
2. «Витягування» молодших бітів і подальше збирання їх у байти;
3. Виконання підрахунку кількості бітів зі значенням «1» у кожному бітовому блоці двійкового представлення;
4. Аналіз результату, підрахування відсотка та виведення його на екран.

Для виконання даного завдання до інтерфейсу користувача особливих вимог не висувається, оскільки всі основні функції будуть виконуватися «програмно». На екран монітора для ознайомлення з процесом виконання завдання достатньо виводити запити на назву файлу для обробки та результат пошуку стеганоконтейнерів в ньому.

1.5.2. Вимоги до інформаційної безпеки

Проблема захисту інформації є багатоплановою, комплексною і охоплює ряд важливих завдань.

Можна виділити вимоги до інформаційної безпеки:

- цілісність даних;
- захист від неавторизованого редагування;
- конфіденційність інформації;
- доступність інформації для всіх авторизованих користувачів.

Надійність роботи розроблюваного програмного забезпечення залежить від надійності операційної системи, під управлінням якої вона буде функціонувати і розроблюваного ПЗ.

Для надійної роботи додатку необхідно:

- використовувати ліцензійне програмне забезпечення на сервері;
- здійснювати захист від вірусів на сервері;

— здійснювати захист від несанкціонованого доступу;

— застосовувати на сервері джерело безперебійного живлення для захисту від перепадів напруги або збоїв у живленні;

— здійснювати контроль даних, що вводяться користувачем.

1.5.3. Вимоги до складу та параметрів технічних засобів

Системні вимоги для застосування даного програмного продукту:

– 1 GB RAM (рекомендується 1.5 GB +);

– 1-2 GB вільного простору на жорсткому диску;

– стандартний GPU з підтримкою DirectX 9.0 та вище;

– роздільна здатність екрану 1024x768 та вище;

– Intel® Pentium® або сумісний, мінімум 1.4 GHz (рекомендується 2.2GHz +);

– переферійні пристрої введення-виведення: клавіатура, миша.

1.5.4. Вимоги до інформаційної та програмної сумісності

Дане програмне забезпечення повинне бути розроблене з використанням наступних програмно-апаратних засобів:

– операційна система Microsoft Windows XP/Vista/7;

– мови програмування C++ за допомогою середовища розробки Microsoft Visual Studio 2017;

Програма повинна являти собою самостійний виконуваний модуль, бути структурована і закоментована.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Запропонований в роботі алгоритм передбачає, що стеганографічним контейнером є растрове зображення, в якому секретні дані приховані в молодших бітах значень кольорових каналів кожного пікселя. Проведений аналіз розподілів бітів у зображеннях, одержаних за допомогою популярних стеганографічних програм, дозволив виділити характерні ознаки, що можуть свідчити про наявність прихованих даних, що було покладено в основу алгоритму.

Програмна реалізація складається з чотирьох основних частин:

1. Зчитування файлу (зображення).
2. «Витягування» молодших бітів і подальше збирання їх у байти;
3. Виконання підрахунку кількості бітів зі значенням «1» у кожному бітовому блоці двійкового представлення;
4. Аналіз результату, підрахування відсотка та виведення його на екран.

Розроблена програма виявляє приховане повідомлення в растрових 24-х бітних зображеннях BMP-формату та виявляє ймовірність наявності стегоданих в діапазоні від 0 до 100%, з найбільш точним результатом.

2.2. Опис застосованих математичних методів

2.2.1. Метод сигнатурного аналізу

Найпростішими методами аналізу контейнерів-зображень є візуальні методи. Візуальні методи намагаються виявити існування стеганографічного вкраплення за допомогою візуального контролю (неозброєним оком) або за допомогою автоматизованих процесів. Візуальний контроль за допомогою неозброєного ока матиме успіх, коли стеганографічні дані вкраплені у

однотонні фрагменти зображення. Автоматизовані комп'ютерні додатки дозволяють розкласти зображення на його індивідуальні бітові площини. Бітова площина складається з одного біту пам'яті для кожного пікселя в зображенні, і є типовим місцем зберігання інформації, прихованої за допомогою стеганопрограм. Будь-який незвичний зовнішній вигляд у відображенні площини молодшого двійкового розряду буде, ймовірно, означати існування вкраплених стеганографічних даних.

Методи сигнатурного аналізу зображення дозволяють відшукати бітові послідовності, специфічні для певних програм стеганоприховання. Стеганоаналіз на основі сигнатур може вимагати дуже багато часу, тому що спочатку потрібно розпізнати сигнатуру для певної стеганографічної програми з великої вибірки файлів, які були вкраплені з її допомогою. Крім того, повинні бути використані автоматизовані процеси для пошуку всіх потенційних файлів-контейнерів для цієї певної сигнатури.

До переваг сигнатурних методів відноситься можливість отримання результату, який однозначно характеризує застосовану для приховання даних стеганосистему. Основним недоліком є невелике (менш 10%) число стеганопрограм, що залишають у контейнерах свої сигнатури.

2.2.2. Статистичні методи аналізу

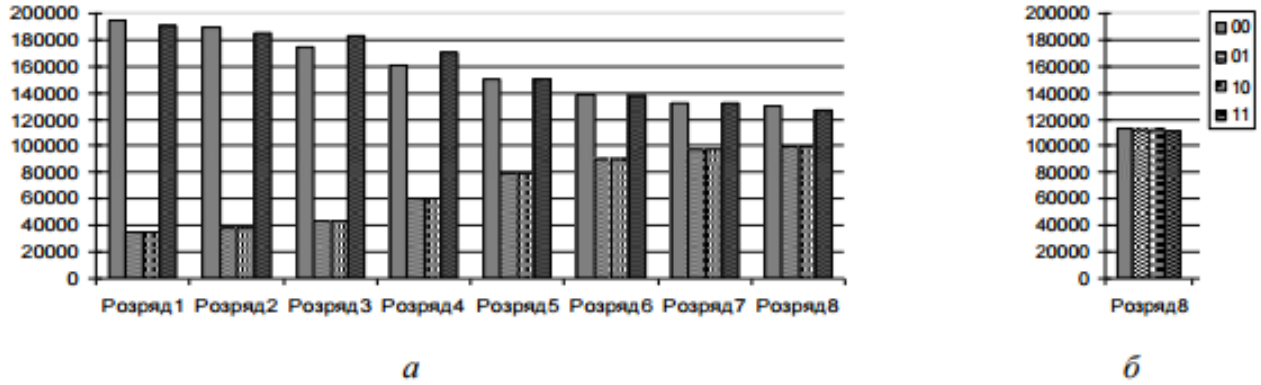
Статистичні методи аналізу намагаються виявити найменші зміни в статистичній поведінці файлу, викликані вкрапленою стеганографією. Суть статистичних методів полягає в оцінюванні ймовірності існування стеганографічного приховання з невідомою стеганосистемою на основі критерію оцінки наближення досліджуваного контейнера до «природного».

2.2.3. Метод оцінки числа переходів значень молодших бітів у сусідніх елементах контейнера

У методі використовується знання, що між молодшими бітами сусідніх елементів, і між ними і іншими бітами природних контейнерів є кореляційні зв'язки. При аналізі графічних файлів формату BMP в якості елементів послідовності, яка аналізується, вибираються найменш значущі біти (НЗБ) пікселів, що розташовуються поруч, колірних складових зображення. При дослідженні файлів формату JPEG – молодші біти сусідніх коефіцієнтів ДКП, відмінних від 0 і 1.

Під «переходом» розуміють перехід значення і-го елемента послідовності в значення і □1 елемента послідовності x, $i = 1,2,...,n-1$, n – довжина послідовності. Оскільки послідовності є двійковими, то аналізуються чотири види переходів: з 0 в 0, з 0 в 1, з 1 в 0 і з 1 в 1. За отриманими результатами будується гістограма [3]. Для кожного розряду перший стовпець гістограми показує число переходів у потоці НЗБ із 0 в 0, другий стовпець – з 0 в 1, третій стовпець – з 1 в 0, четвертий стовпець – з 1 в 1.

Для пустого контейнера і контейнера, що містить вкраплену інформацію, число переходів у потоці НЗБ буде різним. Розподіл НЗБ стеганоконтейнера має, як правило, випадковий характер. Відповідно число переходів у потоці НЗБ для всіх станів буде приблизно однаковим, що не властиво пустому контейнеру (рис. 2.1, а, б).



а *б*

Рис. 2.1. Гістограма частот переходів бітових значень: а – пустого контейнера, б – стеганоконтейнера (восьмий розряд контейнера, в який були внесені зміни)

Статистичний критерій для оцінки частот переходів бітових значень [3]: файл, що аналізується розбивається на K блоків однакової довжини і вибирається деяке порогове значення h_M . Обчислюються значення статистик:

$$M_j = \left[\frac{(\mu_{00} - \mu_{01})^2}{2} + \frac{(\mu_{11} - \mu_{10})^2}{2} \right], j = 1,2,...K, \quad (2.1)$$

де $\mu_{i,j}$ - кількість переходів у потоці НБЗ з і в j (i,j=0,1). У випадку якщо $M_j<h$, вважається, що в j-ому блоці міститься прихована інформація.

2.2.4. Метод оцінки частот

Метод оцінки частот появи k - бітових серій у потоці НЗБ елементів контейнера. Метод дозволяє оцінити рівномірність розподілу елементів в послідовності, яка досліджується, на основі аналізу частоти появи нулів і одиниць, і серій, що складаються з k бітів [4]. У бітовому поданні послідовності x , що досліджується, підраховується скільки разів зустрічаються нулі і одиниці (k = 1), серії-двійки (00, 01, 10, 11: k = 2), серії-трійки (000, 001, 010, 011, 100, 101, 110, 111: k = 3) і т.д. На основі результатів будується гістограма.

Для незаповнених BMP і JPEG зображень не є характерним, щоб значення частот всіх компонентів знаходились досить близько (рис. 2.2., а). При вкрапленні інформації, значення частот зближаються (рис. 2.2, б). Цей факт використовується при аналізі.

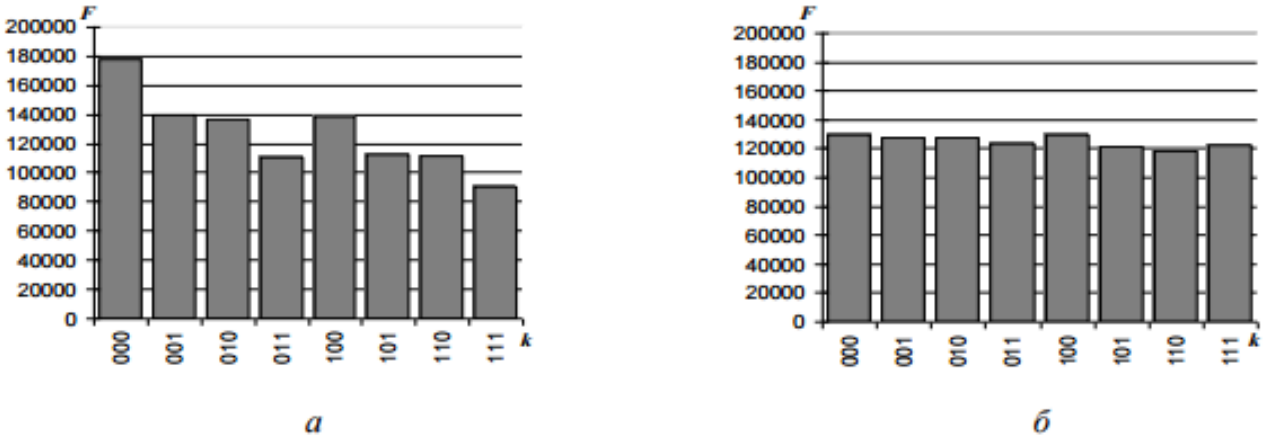


Рис. 2.2. . Гістограма частот серії-трійки (k = 3) у потоці НЗБ:
а – пустого контейнера, б – стеганоконтейнера

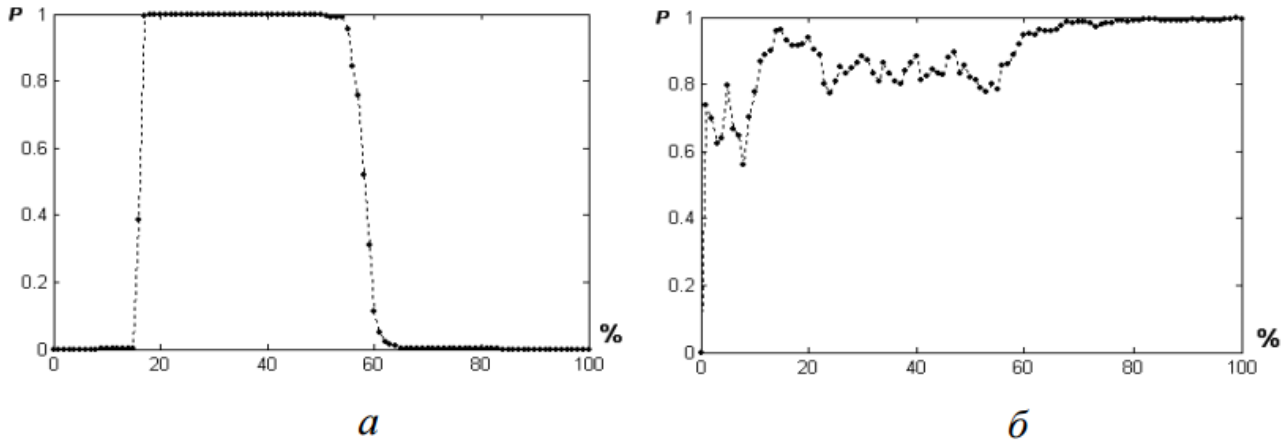
Результати роботи методу залежать від стеганографічного перетворення і від обсягу даних, що приховуються. Як правило, виявлення факту приховання здійснимо при заповненні контейнера на 60% і вище.

2.2.5. Метод аналізу розподілу пар значень на основі критерію X^2□□□□□

У методі використовується аналіз гістограми, отриманої за елементами зображення і оцінка розподілу пар значень цієї гістограми [1]. Для BMP файлів пари значень формуються значеннями пікселів зображення, для JPEG – квантованими коефіцієнтами дискретного косинусного перетворення, які відрізняються за молодшим бітом. Молодші біти зображень не є випадковими. Частоти двох сусідніх елементів контейнера мають перебувати досить далеко від значення частоти середнього арифметичного цих елементів. В «пустому»

зображенні ситуація, коли частоти елементів зі значеннями 2N і 2N □1 близькі за значенням, зустрічається досить рідко. При вкрапленні інформації дані частоти зближаються або стають рівними. Ідея атаки X^2□полягає в пошуку цих близьких значень і підрахунку ймовірності вкраплення на основі того, як близько розташовуються значення частот парних і непарних елементів аналізованого контейнера. Особливістю алгоритму є послідовний аналіз всього зображення і, відповідно, накопичення частот елементів.

Метод X^2 є універсальним, оскільки підходить для аналізу зображень, в які інформація вкраплювалася за допомогою різних стеганографічних програм. Однак результати роботи методу за критерієм X^2 значною мірою залежать від методу приховання даних. При послідовній заміні НЗБ елементів контейнера і вкрапленні повідомлення з заповненням метод виявляє наявність прихованих даних (рис. 2.3, а, б), а при псевдовипадковому виборі молодших бітів (розподіленому вкрапленні) метод не спрацьовує.



а *б*

Рис. 2.3. Ймовірність вкраплення за критерієм X^2 : а – послідовне вкраплення, б – вкраплення з заповненням

2.2.6. Метод аналізу гістограм

Метод аналізу гістограм, побудованих за частотами елементів зображення дозволяє оцінити рівномірність розподілу елементів зображення, що

аналізується, а також визначити частоту появи конкретного елемента.

Якщо частоти двох сусідніх елементів BMP зображення близькі за значенням і/або розташовані з різницею в одиницю (наслідок використання класичного методу НЗБ), то контейнер містить приховані дані (рис. 6, б). В іншому випадку контейнер вважається пустим (рис. 2.4, а) [4].

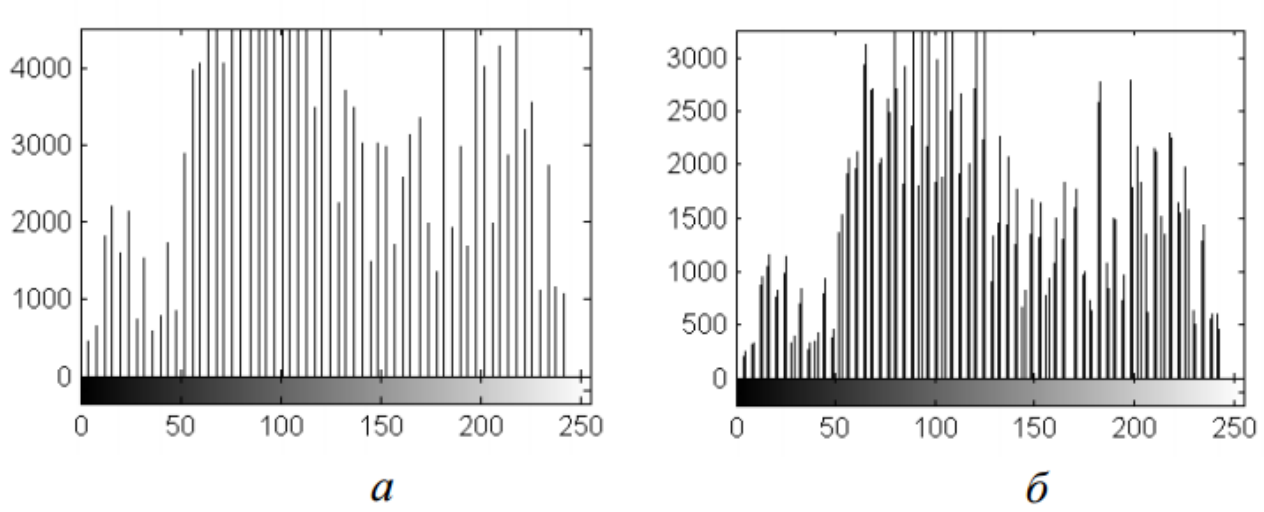


Рис. 2.4. Гістограма частот пікселів: а – вихідного BMP зображення, б – BMP зображення, що містить приховану інформацію

2.2.7. Метод аналізу розподілу елементів зображення на площині

Метод аналізу розподілу елементів зображення на площині призначений для визначення залежностей між елементами послідовності, що досліджується.

На площину (поле) розміром $(2^R-1)x(2^R-1)$, де R – розрядність елемента послідовності, наносяться точки з координатами $(x_i, x_{i+□1})$, x_i – елементи послідовності x , що досліджується, i=1,2,...,n-1 , n – довжина послідовності [4]. За отриманим результатом проводиться аналіз.

Якщо точки по всьому полю розташовані хаотично, то між елементами послідовності відсутні залежності, що характерно для контейнерів з вкрапленими даними (рис. 2.5, б). У випадку пустого контейнера точки на полі будуть розташовані нерівномірно або утворювати “візерунки” (рис. 2.5, а).

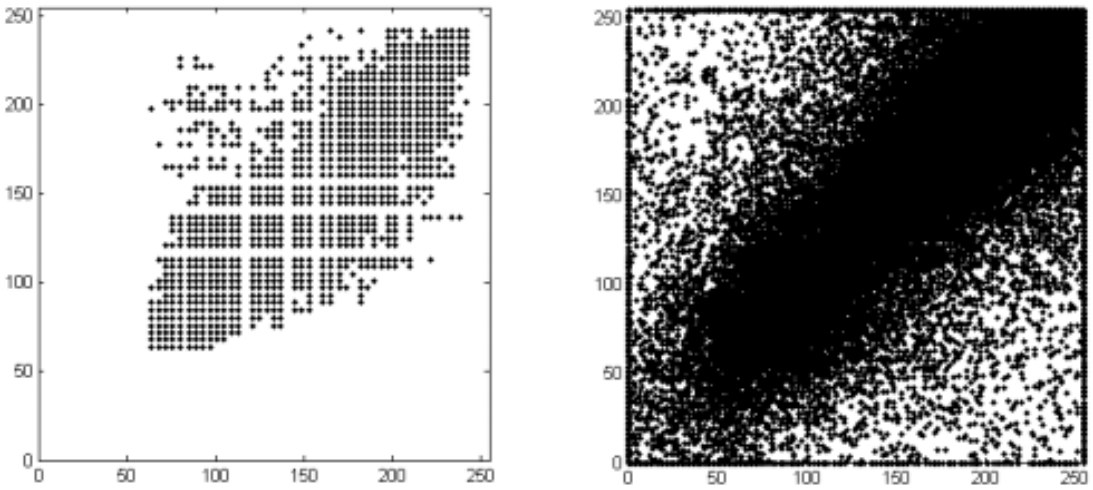


Рис. 2.5. Розподіл на площині елементів: а – вихідного зображення, б – зображення, що містить приховану інформацію

Крім наведених вище, існує багато статистичних методів, заснованих на різних математичних моделях процесу стеганографії. Статистичні методи не є засобом, який дозволяє з 100% надійністю визначати наявність прихованої інформації. Вони дають можливість стеганоаналітику з певною ймовірністю судити про те, використовувалось стеганографічне перетворення чи ні.

2.3. Опис використаної архітектури та шаблонів проектування

Оптимізація проекту на рівні побудови архітектури є основним фактором, який необхідний при створенні високонавантаженого проекту. За допомогою паттерна mvc проходить оптимізація в сторону написання надлишкового коду, робить систему більш гнучкою і легко конфігурованою.

Model-view-controller (MVC, «модель-уявлення-контролер») - схема використання декількох шаблонів проектування, за допомогою яких модель даних програми, призначений для користувача інтерфейс і взаємодія з користувачем розділені на три окремих компонента таким чином, щоб

модифікація одного з компонентів надавала мінімальний вплив на інші. Дана схема проектування часто використовується для побудови архітектурного каркаса, коли переходять від теорії до реалізації в конкретній предметній області (рис.2.6).

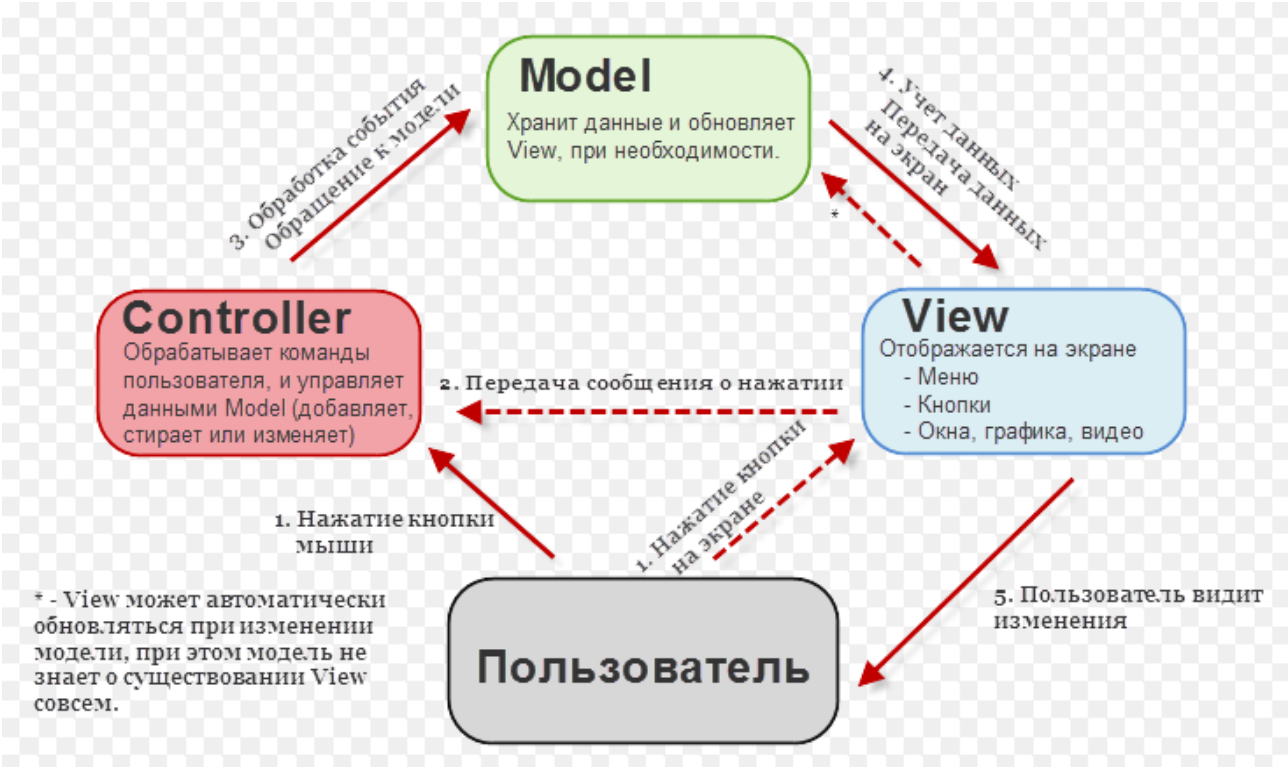


Рис. 2.6. Схема MVC

Ця схема наочно показує односпрямованість потоку інформації в паттерні, а також описує ролі кожного компонента.

Модель використовується для доступу і маніпулювання даними. У більшості випадків модель - це те, що використовується для доступу до сховища даних (бази даних, або файлів даних). Модель надає інтерфейс для пошуку даних, їх створення, модифікації і видалення зі сховища. Модель є посередником між поданням і контролером.

Подання (уявлення) - це те, де дані, отримані від моделі, виводяться в потрібному вигляді. Подання також відповідає за отримання дій від

користувача з тим щоб відправити їх контролеру. Подання відображає кнопку в інтерфейсі, а після її натискання викликає відповідну дію контролера.

Також важливим моментом є те, що уявлення ніколи не працює з «чистими» даними від контролера, тобто, контролер ніколи не працює з поданням в обхід моделі. В процесі взаємодії контролера та подання модель завжди знаходиться між ними.

Контролер - це остання частина зв'язки MVC. Завданням контролера є отримання даних від користувача і маніпуляція моделлю. Саме контролер і тільки він є тією частиною системи, яка взаємодіє з користувачем.

2.4. Опис використаних технологій та мов програмування

C++ компільована, статично типізована мова програмування загального призначення.

Підтримує такі парадигми програмування як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування, забезпечує модульність, роздільну компіляцію, обробку винятків, абстракцію даних, оголошення типів (класів) об'єктів, віртуальні функції. Стандартна бібліотека включає, в тому числі, загальновживані контейнери і алгоритми. C ++ поєднує властивості як високорівневих, так і низькорівневих мов. У порівнянні з її попередником мовою C, - найбільшу увагу приділено підтримці об'єктно-орієнтованого і узагальненого програмування.

C++ широко використовується для розробки програмного забезпечення, будучи однією з найпопулярніших мов програмування. Область її застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високопродуктивних серверів, а також розважальних програм. Існує безліч реалізацій мови C++, як безкоштовних, так і комерційних і для різних платформ [2].

C++ мова, що складалася еволюційно. Кожен елемент C++ запозичувався з інших мов окремо і незалежно від інших елементів (ніщо з запропонованого

C++ за всю історію його розвитку не було нововведенням в Computer Science), що зробило мову надзвичайно складною, з безліччю дублюючих і взаємно суперечливих елементів, блоки яких засновані на різних формальних базах.

Критики C++ не протиставляють їй який-небудь конкретну мову, а навпаки, стверджують, що для будь-якого випадку застосування C++ завжди існує альтернативний інструментарій, що дозволяє вирішити ту ж задачу більш ефективно і якісно. У свою чергу, прихильники C++ вважають некоректним порівнювати різні аспекти C++ з абсолютно різними мовами, так як загальний набір засобів і можливостей C++ істотно ширше, ніж в більшості мов, з якими проводиться порівняння, і сама по собі широта можливостей , на їх погляд, є вагомим виправданням недосконалості кожної окремо взятої можливості. Більш того, на їхню думку, висока сумісність з C і є однією з важливих характеристик мови, і тому всі недоліки C++ виправдані перевагами, наданими цій сумісністю.

Переваги:

- висока сумісність з мовою C;
- обчислювальна продуктивність;
- підтримка різних стилів програмування: структурне, об'єктно-орієнтоване, узагальнене програмування, функціональне програмування;
- автоматичний виклик деструкторів об'єктів (в порядку зворотному виклику конструкторів) спрощує і підвищує надійність управління пам'яттю і іншими ресурсами (відкритими файлами, мережевими з'єднаннями, тощо);
- перевантаження операторів;
- шаблони (дають можливість побудови узагальнених контейнерів і алгоритмів для різних типів даних);
- можливість розширення мови для підтримки парадигм, які не підтримуються компіляторами безпосередньо
- доступність (для C ++ існує величезна кількість навчальної літератури, перекладеної на різні мови).

Недоліки:

– погано продуманий синтаксис звужує спектр застосування мови;

– мова не містить багатьох важливих можливостей;

– мова містить небезпечні можливості;

– продуктивність праці програмістів на мові виявляється невиправдано низька;

– громіздкість синтаксису [3].

При розробці ІС використано бітову бібліотеку «bitmap_image.hpp» (рис. 2.7). Бітова бібліотека С ++ складається з простих, надійних, оптимізованих та портативних процедур обробки для формату 24-бітових піксельних бітових зображень. Бібліотека є безкоштовною.

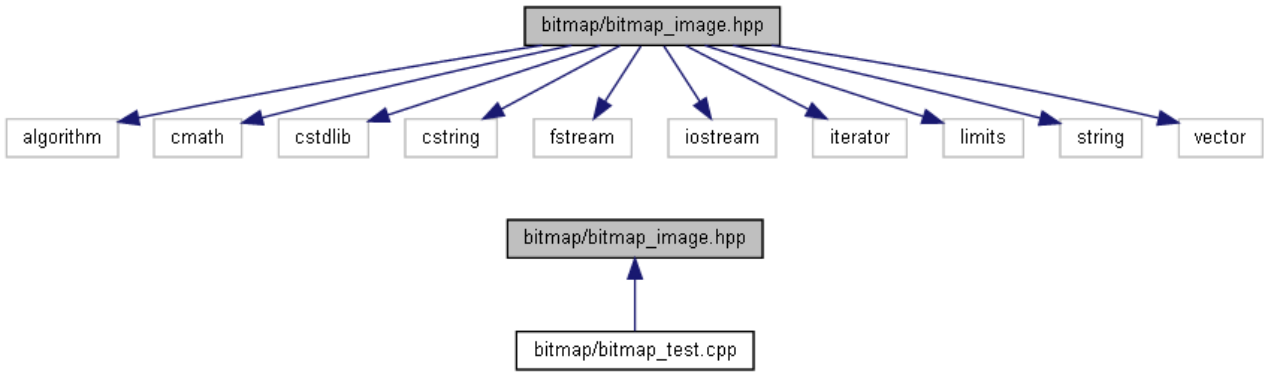


Рис. 2.7. Структура бібліотеки «bitmap_image.hpp»

Бібліотека “bitmap_image.hpp”має такі можливості:

- читання / запис 24-бітних зображень;
- редагування пакета на рівні пікселів, рядків або стовпців;
- кольорові перетворення (RGB, YCbCr) в байтах;
- високооптимізований підпроект і примірник (зміна розміру);
- різні кольорові карти;
- формування текстури;
- інтерфейс графіка;
- декартове полотно та відповідний інтерфейс малюнків;

– PSNR та порівняння зображень тощо.

2.5. Опис структури програми та алгоритмів її функціонування

2.5.1. Опис використаних алгоритмів та методів стеганографії

2.5.1.1. Опис алгоритмів стеганографії

Комп'ютерна стеганографія - один з основних видів стеганографії. Існують різні методи для приховування інформації:

– метод LSB(Least Significant Bit);
– метод ДКП (дискретне косинусне перетворення);
– метод відлуння;
– статичні методи (особливий метод);
– методи спотворення;
– структурний метод тощо.

Розглянемо деякі з них докладніше.

LSB (Least Significant Bit, найменший значущий біт) - суть цього методу полягає в заміні останніх значущих бітів у контейнері (зображення, аудіо або відеозапису) на біти приховуваного повідомлення. Різниця між порожнім і заповненим контейнерами повинна бути не відчутна для органів сприйняття людини.

Принцип цього методу полягає в наступному: Припустимо, є 8-бітне зображення в градаціях сірого. 00h (00000000b) позначає чорний колір, FFh (11111111b) - білий. Усього є 256 градацій (2^8). Також припустимо, що повідомлення складається з 1 байта - наприклад, 01101011b. При використанні 2 молодших біт в описах пікселів, нам буде потрібно 4 пікселя. Припустимо, вони чорного кольору. Тоді пікселі, що містять приховане повідомлення, будуть виглядати наступним чином: 00000001 00000010 00000010 00000011. Тоді колір пікселів зміниться: першого - на 1/255, другого і третього - на 2/255 і четвертого - на 3/255. Такі градації, мало того що непомітні для людини, можуть взагалі не відобразитися при використанні низькоякісних пристроїв

виведення. В ролі базового контейнера пропонується використовувати файли BMP-зображень високої роздільності з глибиною кольору 24 та 32 біти, таємне зображення може мати розширення .BMP, .GIF, .PNG, .JPEG.

Недоліком методу LSB є чутливість до розміру зображення, тобто чим менший розмір зображення, тим більше будуть відрізнятися два сусідні пікселі, тому пропонується використовувати зображення з великою роздільністю. Також метод «видає себе» при побітовому перегляді зображення, де чітко видно області зображення в які «вбудовано» таємну інформацію. Попри це, метод запису Least Sagnificant Bit є досить популярним, стійким та простим в реалізації.

Підвиди LSB-алгоритмів для растрових зображень без палітри:

— BlindHide (приховування наосліп). Найпростіший алгоритм: дані записують, починаючи з верхнього лівого кута зображення до правого нижнього - піксел за пікселем. Приховані дані програма записує у наймолодших бітах кольорів пікселя. Приховані дані розподіляються у контейнері нерівномірно. Якщо приховані дані не заповнять повністю контейнер, то лише верхня частина зображення буде засміченою.

— HideSeek (заховати-знайти). Цей алгоритм у псевдовипадковий спосіб розподіляє приховане повідомлення у контейнері. Для генерації випадкової послідовності використовує пароль. Дещо «розумніший» алгоритм, але все ж не враховує особливостей зображення-контейнера.

— FilterFirst (попередня фільтрація). Виконує фільтрацію зображення-контейнера - пошук пікселів, у які записуватиметься прихована інформація (для яких зміна наймолодших розрядів буде найменш помітною для ока людини).

— BattleSteg (стеганографія морської битви). Найскладніший і найдосконаліший алгоритм. Спочатку виконує фільтрацію зображення-контейнера, після чого прихована інформація записується у «найкращі місця» контейнера у псевдовипадковий спосіб (подібно, як у HideSeek).

Метод ДКП - зміна величин коефіцієнтів дискретного косинусного перетворення в зображенні.

Дискретне косинусне перетворення (англ. Discrete Cosine Transform, DCT) - одне з ортогональних перетворень. Варіант косинусного перетворення для вектора дійсних чисел. Різновид перетворення Фур'є і , так само як і воно, має зворотне перетворення. Застосовується в алгоритмах стиснення інформації з втратами, наприклад, MPEG і JPEG.

Графічне зображення можна розглядати як сукупність просторових хвиль, причому осі X і Y збігаються з шириною і висотою зображення, а по осі Z відкладається значення кольору відповідного пікселя зображення. Дискретне косинусне перетворення дозволяє переходити від просторового уявлення зображення до його спектрального подання і назад. Впливаючи на спектральне подання зображення, що складається з «гармонік», тобто, відкидаючи найменш значущі з них, можна балансувати між якістю відтворення і ступенем стиснення.[1]

Переваги цього методу в тому, що він стійкий до стискання зображення. Малюнок ділять на блоки (8x8 пікселей), після чого для обраного блоку та двох певних коефіцієнтів ДКП шифрується секретна інформація. Коефіцієнти - косинус-функції, відповідні середнім частотам.

Інші методи приховування інформації в графічних файлах орієнтовані на формати файлів з втратою, наприклад, JPEG. На відміну від LSB вони більш стійкі до геометричних перетворень. Це виходить за рахунок варіювання в широкому діапазоні якості зображення, що призводить до неможливості визначення джерела зображення.

2.5.1.2. Реалізація методу LSB

Значення LSB - найменш значущий біт. У контейнері (зображення, аудіо файли тощо) два молодших біта (наприклад, 11111100 → 00) замінюють на потрібні значення. Приклад зображення, в якому наявні приховані дані наведено на рис. 2.8:

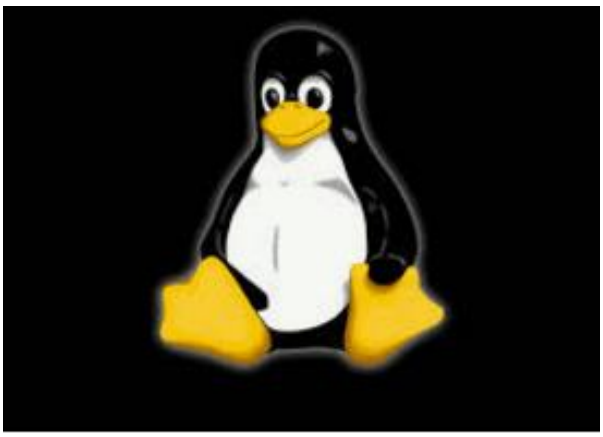


Рис. 2.8. Зображення з скритим текстовим повідомленням

При цьому різниця між оригінальним та зашифрованим файлом для людини не помітна. Недолік цього методу в тому, що він нестійкий до всіх видів атак на стеганографію.

Для наочного прикладу візьмемо 8-бітне зображення формату BMP. Перші два байта заголовка - це сигнатура BM, далі в подвійному слові записаний розмір файлу в байтах, наступні 4 байти зарезервовані й повинні містити нулі і, нарешті, в ще одному подвійному слові записано зміщення від початку файлу, до власне байтів зображення. У 24-бітному bmp-файлі кожен піксель кодуються трьома байтами RGB(палітра Red, Green, Blue).

В кожному з каналів замінюємо 2 молодших біта → білий колір (255,255,255) [255]→11111111, два молодших біта 11 можемо замінити на 10, 01 або 00, отриманий колір буде виглядати так (254, 253, 252 (змінені молодші біти: 10, 01, 00)). Зрозуміло, що можна замінювати не тільки 2 молодших біти, а і будь-яку їх кількість. Тут є наступна закономірність: чим більшу кількість бітів ми міняємо, тим більший обсяг інформації ми можемо заховати, але тим більші підозри в оригінальному документі це викличе.

Приклад алгоритму, в якому записуються текстові данні та замінюються два молодших біта наведено на рис. 2.9:

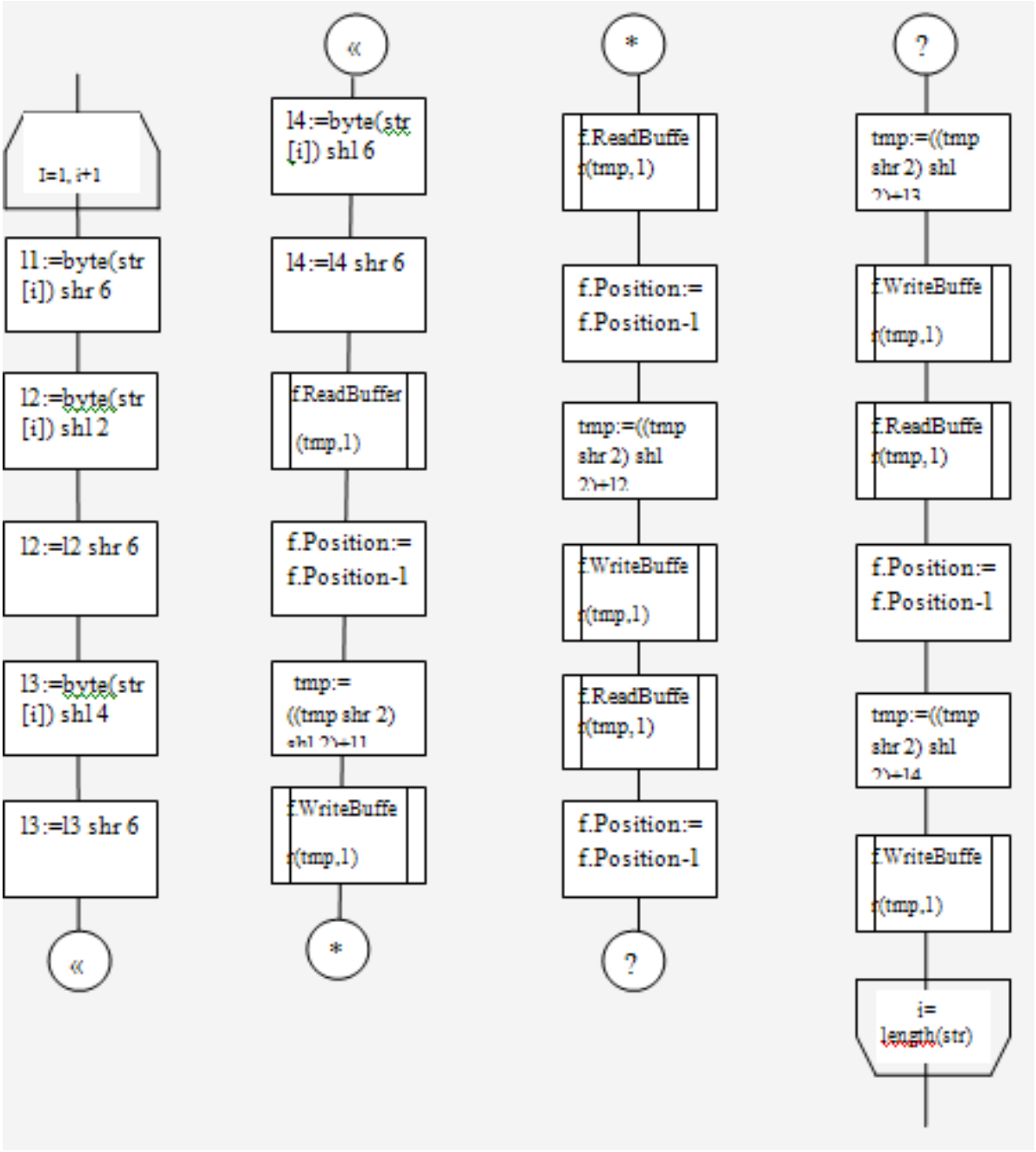


Рис. 2.9. Алгоритм запису текстових даних та заміни бітів

В даному алгоритмі ми зрушуємо значення біта на потрібну нам кількість та зчитуємо за допомогою наступного алгоритму (рис. 2.10):

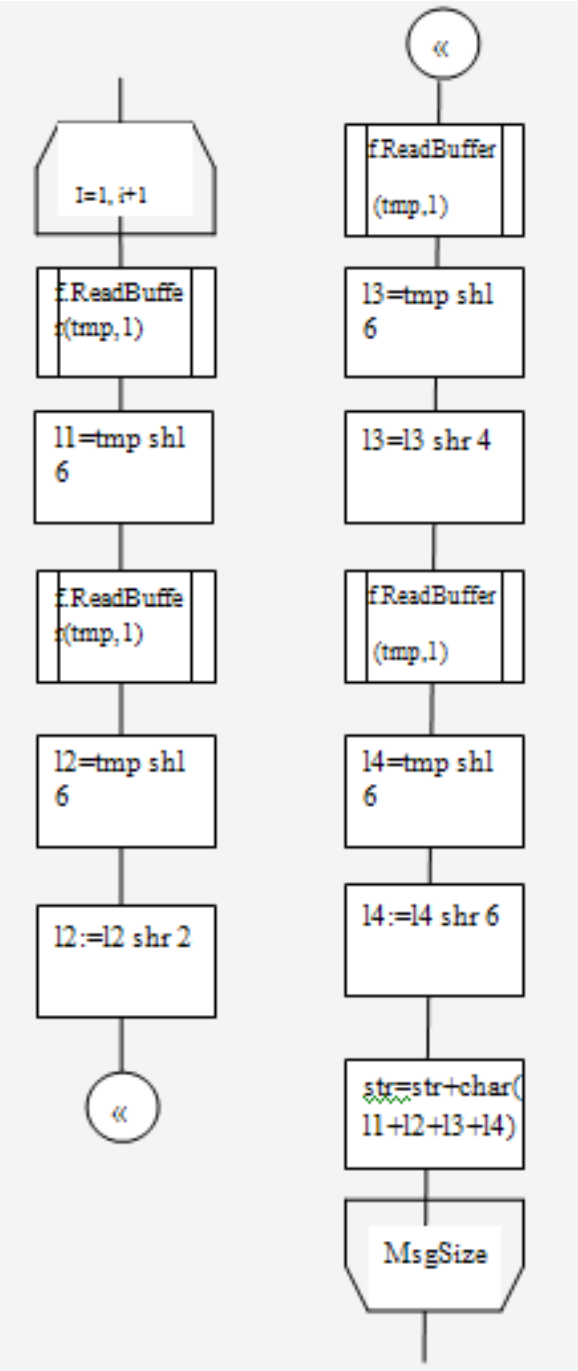


Рис. 2.10. Алгоритм зчитування бітів

За алгоритм сортування розглянемо стандартний алгоритм бульбашкового сортування одновимірного масиву (рис. 2.11).

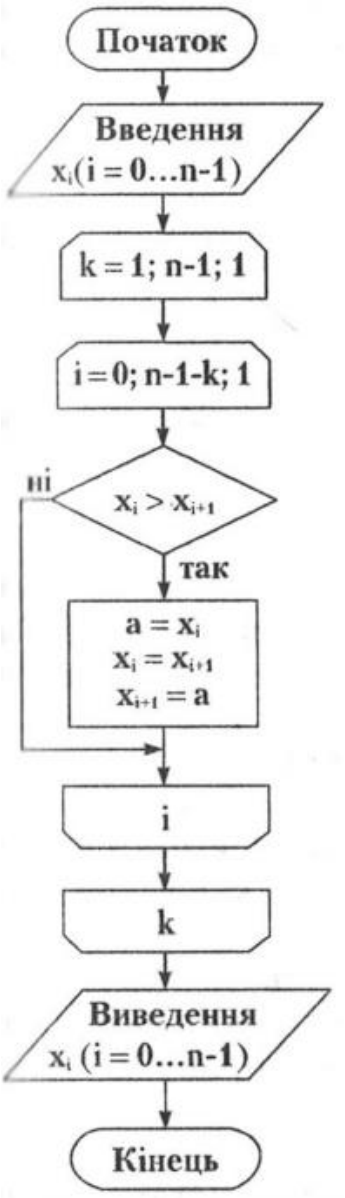


Рис. 2.11. Алгоритм бульбашкового сортування одновимірного масиву

Сортування бульбашкою - найпростіший алгоритм сортування, застосовуваний чисто для навчальних цілей. Практичного застосування цього алгоритму немає, так як він не ефективний, особливо якщо необхідно впорядкувати масив великого розміру. До плюсів сортування бульбашкою відноситься простота реалізації алгоритму.

Алгоритм сортування бульбашкою зводиться до повторення проходів за елементами сортованого масиву. Прохід по елементам масиву виконує внутрішній цикл. За кожен прохід порівнюються два сусідні елементи, і якщо порядок невірний елементи міняються місцями. Зовнішній цикл буде

працювати до тих пір, поки маса не буде відсортований. Таким чином зовнішній цикл контролює кількість спрацьовувань внутрішнього циклу Коли при черговому проході за елементами масиву не буде здійснено жодної перестановки, то масив буде вважатися відсортованим.

2.5.1.3. Опис етапів та методів стегоаналізу

Оскільки стеганографія дуже поширена у сучасному цифровому світі, то не менш важливим фактором є стійкість стеганографічних систем - ймовірність пропуску (тобто часткової відсутності виявлення стегосистеми, яка була запропонована для аналізу) і ймовірність помилкового виявлення (коли стегосистеми помилково виявляються при їх дійсній відсутності). Якщо ці дві умови збалансовані, то таку систему можна вважати стійкою. Не зважаючи на складність виявлення таких систем, вони все одно будуть вразливі для деяких видів стеганографічних атак (один з етапів – стегоаналіз), які побудовані на тому, що всі алгоритми вбудовування так чи інакше вносять в стегограми спотворення щодо використаних контейнерів.

Таким чином стегоаналіз складається з трьох етапів.

У першому етапі вибирається метод стеганографіії, за допомогою якого могла бути прихована інформація, після чого структурується повідомлення у вигляді контейнера та складається уявлення щодо можливих способів додавання стегознаку у повідомлення конкретним методом.

Другий етап - це виявлення можливих атак для досліджуваного елементу (повідомлення), тобто видозмін контейнера.

Третім етапом є сам стегоаналіз, в ході якого контейнер піддається атакам. Після цього робиться висновок про наявність чи відсутність стегознаків у повідомленні.

Якщо у ході атаки вдалося виявити наявність прихованої інформації, то таку атаку називають успішною. Слід зауважити, що стегоаналіз дає лише ймовірність результату, а не точну відповідь.

Розглянемо декілька методів стеганоаналізу. Найпростішими методами вважаються аналізи контейнерів-зображень - візуальні методи. Очевидно, що вони намагаються виявити стего за допомогою візуального контролю або за допомогою автоматизованих процесів. Візуальний контроль буде ефективний, коли розглядаються однотонні фрагменти зображення. Автоматизовані комп'ютерні додатки дають можливість розкласти зображення на його індивідуальні бітові площини. Бітова площина складається з одного біта пам'яті для кожного пікселя у зображенні, і є типовим місцем зберігання інформації, прихованої за допомогою стеганопрограм. Будь-який незвичний зовнішній вигляд у відображенні площини молодшого двійкового розряду буде, ймовірно, означати існування вкраплених стеганографічних даних.

Статистичні методи аналізу намагаються виявити найменші зміни у поведінці файлу, викликані випливом стеганографії на даний контейнер. Ідея статистичних методів полягає в оцінюванні ймовірності існування стеганографічного приховування з невідомою стеганосистемою на основі критерію оцінки наближення досліджуваного повідомлення до істинного.

Методом структурного аналізу називають метод дописування даних у кінець контейнера завдяки використанню системи маркерів. Усі стандартні програми перегляду, доходячи до маркера «кінець зображення», припиняють роботу, і прихована інформація залишається непізнаною. Та метод структурного аналізу дозволяє виявити її. Розпізнати зміни у форматі файлу даних можливо за допомогою шістнадцятирічного редактора. Використовуючи інформацію заголовка, можна виявити «кінець зображення», а отже, визначити місце розташування позиції прихованих даних.

Методи сигнатурного аналізу зображення дають змогу відшукати бітові послідовності, специфічні для деяких програм стеганоприховування. Стеганоаналіз на основі сигнатур може вимагати дуже багато часу, тому що спочатку потрібно розпізнати сигнатуру для певної стеганографічної програми з великої вибірки файлів, які були прихованні за її допомогою.

2.5.2. Частотний аналіз зображень, що містять стеганографічні дані

Для подальшої роботи з зображенням їх слід детально проаналізувати.

Отже, головним завданням є проаналізувати растрові зображення BMP-формату та порівняти їх результати. Попередньо були обрані 9 зображень: зображення з насиченими кольорами, однотонні зображення, чорно-білі зображення (без стеганографічних даних) та зображення із стеганографічними даними (рис. 2.12).

Рис 2.12. Приклади зображень, які використовувалися для аналізу

Оскільки ми розглядаємо вкраплення інформації за допомогою методу LSB, то слід зображення розбити на пікселі, та «витягти» 2 молодших біта з кожного байта зображення при використанні колірної моделі RGB (рис. 2.13). Для реалізації цього аналізу (як і самого програмного забезпечення) буде використано мову програмування C++ та бібліотеку “bitmap_image.hpp ”:

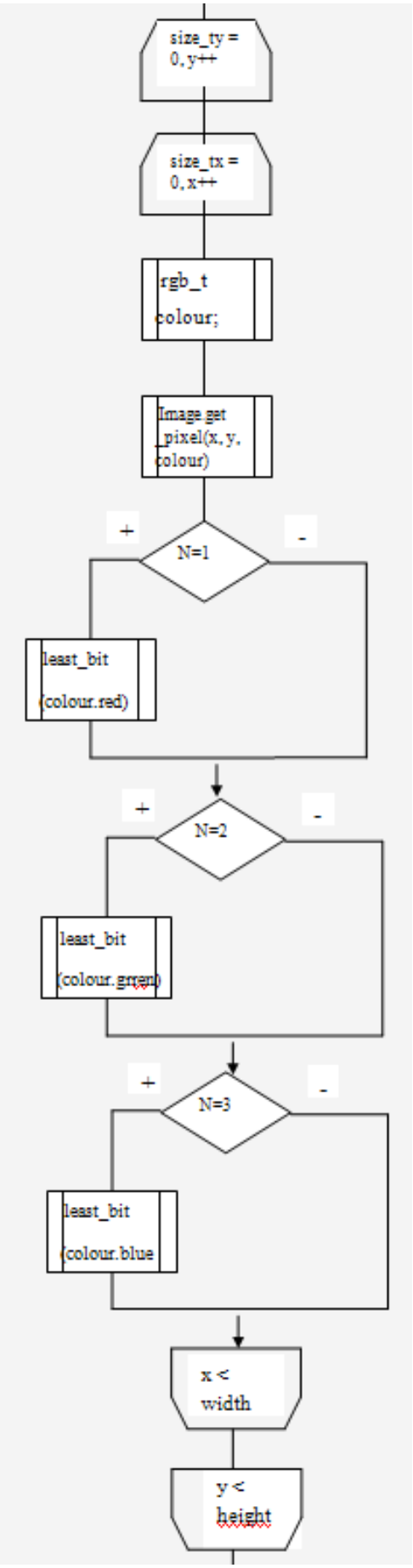


Рис. 2.13. Алгоритм розбиття моделі RGB

Розглянемо функцію “least_bit”, яка виводить два молодших біта:

```
void least_bit(unsigned char k)
{
   int i=(int)k;
   if (i==0) cout<<"00"<<" ";
   else if (i==1) cout<<"01"<<" ";
   else
   {
   int second=i%2;
   int first=(i/2)%2;
   cout<<first<<second<<" ";
   }
}
```

Результат такого аналізу (з компонентом червоного кольору) над зашифрованим зображенням (2.14) наведено на рис 2.15:

Рис 2.14. Зображення, що має зашифроване послання

Рис. 2.15. Зашифрований малюнок розбито на молодші біти

Як ми бачимо, на однотонній ділянці білого кольору молодші біти відрізняються від «природного» зображення, де вони повинні складатися з «1» (“11 11 11 …”) (рис 2.16):

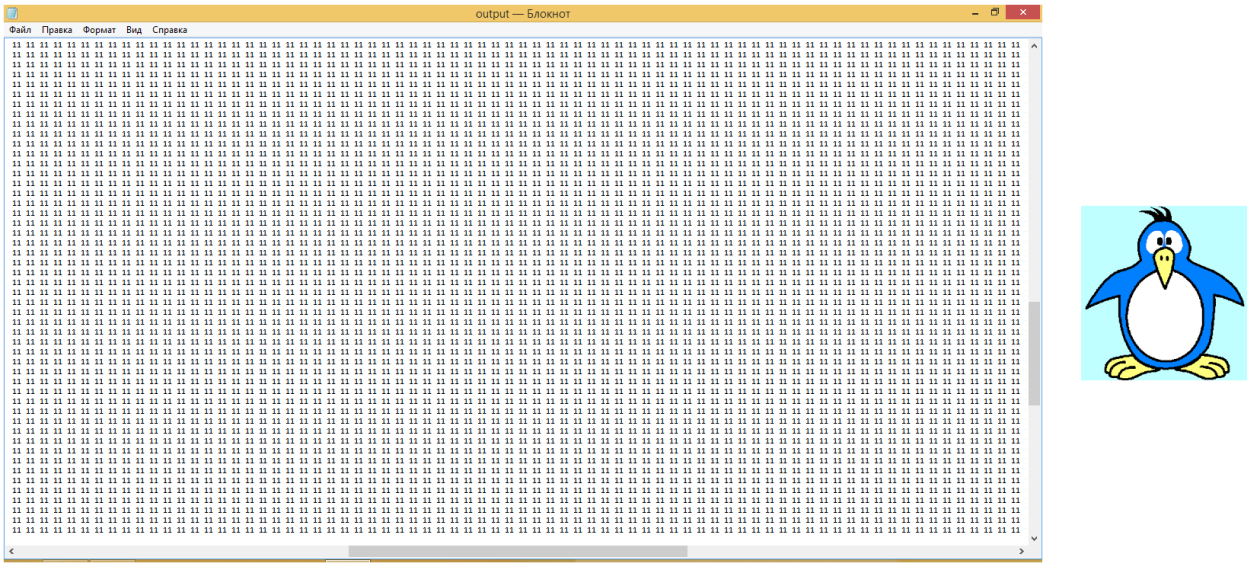


Рис 2.16. Ділянка тіла пінгвіна білого кольору

Після цього зберемо молодші біти (з компонентами червоного, зеленого та синього кольорів) у байти (00 01 10 00 → 00011000) та підрахуємо кількість повторів кожного значення байту (рис. 2.17):

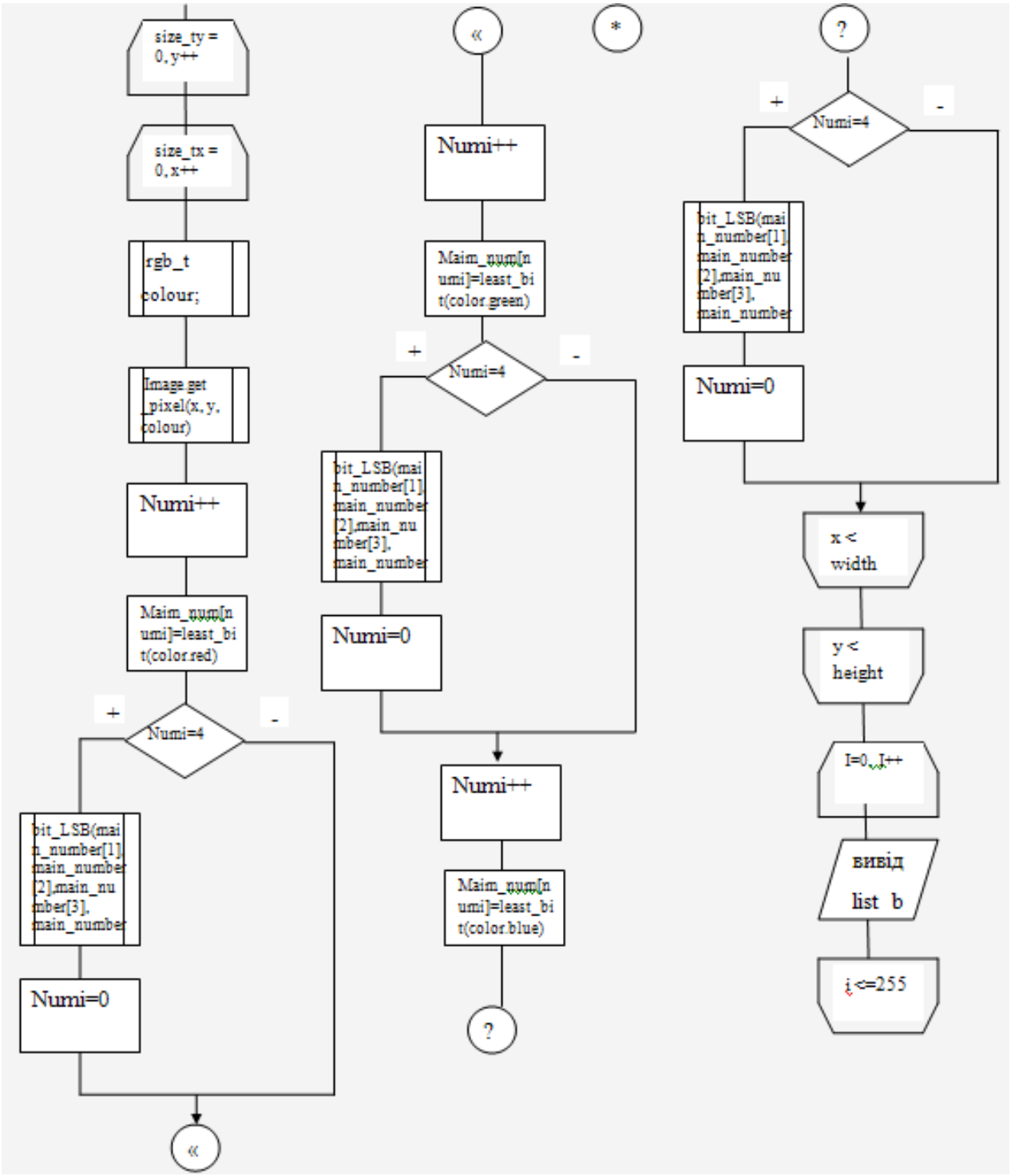


Рис. 2.17. Алгоритм підрахунку повторів

Розглянемо функцію “bit_LSB” для збирання молодших бітів та складання їх у байт:

```
int bit_LSB(string fis,string sec,string thi,string fou)
{
```

```
char k[100];
int a[8];
k[0] = fis[0];
k[1] = fis[1];
k[2] = sec[0];
k[3] = sec[1];
k[4] = thi[0];
k[5] = thi[1];
k[6] = fou[0];
k[7] = fou[1];
a[0] = static_cast<int>(k[0])-48;
a[1] = static_cast<int>(k[1])-48;
a[2] = static_cast<int>(k[2])-48;
a[3] = static_cast<int>(k[3])-48;
a[4] = static_cast<int>(k[4])-48;
a[5] = static_cast<int>(k[5])-48;
a[6] = static_cast<int>(k[6])-48;
a[7] = static_cast<int>(k[7])-48;
return pow(2,7)*a[0]+pow(2,6)*a[1]+pow(2,5)*a[2]+pow(2,4)*a[3]+
pow(2,3)*a[4]+pow(2,2)*a[5]+pow(2,1)*a[6]+a[7]; }
```

Результат такого досліду (з компонентом червоного кольору) над зашифрованим зображенням наведено на рис. 2.18:

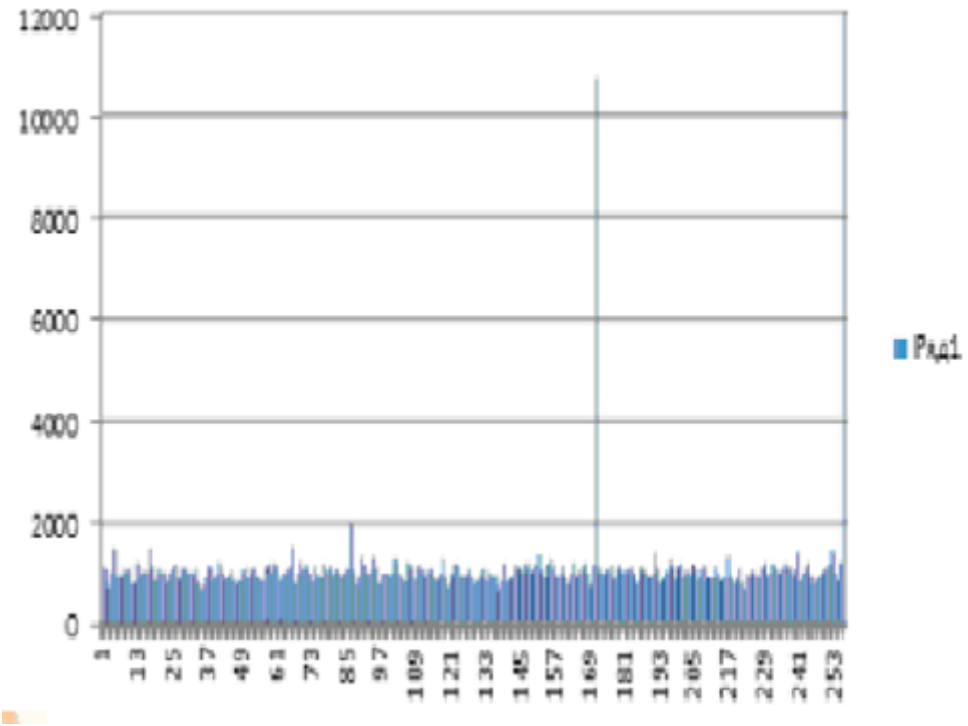


Рис 2.18. Зашифроване зображення та гістограма аналізу до нього

На графіку по осі x відкладено значення байту, а по осі y – кількість байт, що мають таке значення.

Як ми бачимо, наявний пік на значенні «169», що й говорить про наявність стеганографічних даних у зображенні.

2.5.3. Опис запропонованого алгоритму

Методів виявлення наявності стеганографічних даних багато, але існує недостатня кількість програмних реалізацій цих методів. В своїй роботі пропонуємо алгоритм для виявлення наявності схованого повідомлення англійською мовою в растрових зображеннях BMP-формату.

Даний алгоритм може бути адаптований для більшості мов програмування:

1. «Витягуємо» два молодших біти з кожного пікселя (з компонентів червоного, зеленого та синього кольорів).

2. Збираємо молодші біти у байти (4 пари для одного байту) тільки з червоним компонентом, тільки з синім компонентом, тільки з синім

компонентом, і з усіма по черзі (R,G,B R,G,B …) (в підсумку маємо 4 варіанти для подальшого аналізу).

3. Підраховуємо кількість одиниць у кожному розряді.

4. Робимо сортування розрядів за кількістю від максимального до мінімального значення.

5. Підраховуємо відсоток вірогідної наявності в зображенні повідомлення англійською мовою (стеганографічних даних) за таким принципом:

– порядок відсортованих розрядів (еталонний порядок → 5, 6, 2, 0, 1, 3, 4,7);

– кількість одиниць у 7 розряді (еталонна кількість → 0);

–різниця в кількості одиниць у сусідніх розрядах за еталонним порядком (5 з 6, 6 з 2, 2 з 0, 0 з 1, 1 з 3, 3 з 4, 4 з 7) (еталонна різниця → 5% від кількості отриманих байтів).

6. Виводимо найбільший відсоток з 4 варіантів (вірогідніше він буде найбільш точним).

В програмі використовується ряд функцій. Одна з них “bit_LSB” створена власноруч. Функція дозволяє збирати молодші біти в байти для подальшого їх перетворення й виводить число у десятковому форматі.

За алгоритм сортування був взятий стандартний алгоритм бульбашкового сортування одновимірного масиву.

2.5.4. Опис структури системи

Програмна реалізація складається з чотирьох основних частин:

1. Зчитування файлу (зображення).
2. «Витягування» молодших бітів і подальше збирання їх у байти;
3. Виконання підрахунку кількості бітів зі значенням «1» у кожному бітовому блоці двійкового представлення;
4. Аналіз результату, підрахування відсотка та виведення його на екран.

Схема реалізації програми наведена на рис. 2.19.

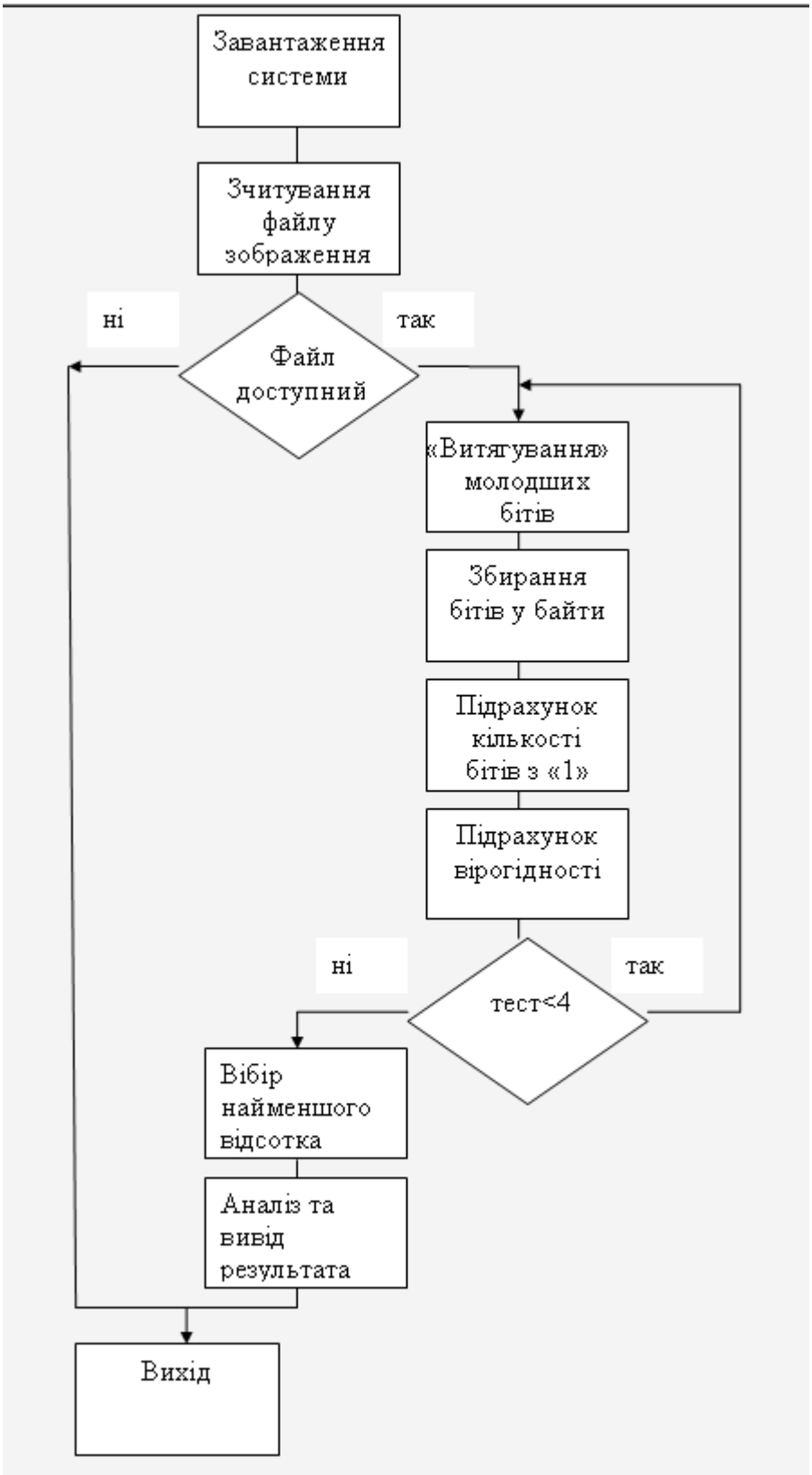


Рис. 2.19. Схема програмної реалізації системи

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення працює з прихованим повідомленням в растрових 24-х бітних зображеннях BMP-формату, які розглядаються як вхідні дані програми та міститься в файлі з відповідним розширенням *.bmp.

Вихідними даними системи є виявлена ймовірність наявності стегоданих в діапазоні від 0 до 100%, що виводиться у текстовому повідомлення на екран монітора користувача.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки даного програмного забезпечення використано ПК з наступними параметрами:

– процесор із тактовою частотою 1 ГГц– 32-розрядний;

– оперативна пам’ять 1 гігабайт (ГБ);

– графічний пристрій із підтримкою DirectX 9 і драйвером WDDM 1.0 або новішим.

2.7.2. Використані програмні засоби

Запропонована програма написана на об'єктно-орієнтованій мові програмування з безпечною системою типізації C++. Вибір був зроблений на користь даної мови, оскільки вона має інтеграцію з .NET Framework, який, в свою чергу, дає можливість, не зменшуючи швидкість розробки, реалізувати заданий алгоритм. До того ж, привабливою є досить нова кросплатформова технологія .NET Core та зрозуміла, але, водночас, потужна об’єктно-орієнтована сторона мови.

Розробка велася в інтегрованому середовищі розробки Microsoft Visual Studio 2017 Community.

2.7.3. Виклик та завантаження програми

Для встановки програми на ПК потрібно папку MAIN скопіювати на потрібний комп’ютер.

Для її правильного використання, необхідно:

1. Перемістити необхідне для аналізу зображення BMP-формату у папку "main".

2. Задати даному зображенню ім’я "inpic.bmp".

3. Запустити файл запуску програми "perc_stego.exe".

Якщо програма не видає результату, потрібно перевірити папку на наявність всіх необхідних файлів (perc_stego.exe, bitmap_image.hpp, perc_stego.cpp, perc_stego.o) та перейменованого зображення BMP-формату (inpic.bmp). В разі наявності цих файлів, потрібно запустити програму через "perc_stego.cpp":

– для цього необхідно мати Code::Blocks (вільне багатоплатформове середовище розробки програмного забезпечення);

– після запуску "perc_stego.cpp" натиснути клавішу "F9" або натиснути мишкою "Build"-->"Build and run".

Приклади зображень для тесту можна знайти у папці "Зображення для тесту".

2.7.4. Опис інтерфейсу користувача

Вхід в додаток виконується після завантаження програми. При цьому додаток одразу звертається до файлу з растровим зображення BMP-формату з ім’ям "inpic.bmp" та починає його обробку наступним чином:

1. «Витягує» два молодших біти з кожного пікселя (з компонентів червоного, зеленого та синього кольорів) заданого зображення.

2. Збирає молодші біти у байти (4 пари для одного байту) тільки з червоним компонентом, тільки з синім компонентом, тільки з синім компонентом, і з усіма по черзі (R,G,B R,G,B …) (в підсумку маємо 4 варіанти для подальшого аналізу).

3. Підраховує кількість одиниць у кожному розряді.

4. Робить сортування розрядів за кількістю від максимального до мінімального значення.

5. Підраховує відсоток вірогідної наявності в зображенні повідомлення англійською мовою (стеганографічних даних) за таким принципом:

— порядок відсортованих розрядів (еталонний порядок → 5, 6, 2, 0, 1, 3, 4,7);

— кількість одиниць у 7 розряді (еталонна кількість → 0);

— різниця в кількості одиниць у сусідніх розрядах за еталонним порядком (5 з 6, 6 з 2, 2 з 0, 0 з 1, 1 з 3, 3 з 4, 4 з 7) (еталонна різниця → 5% від кількості отриманих байтів).

6. Виводить найбільший відсоток з 4 варіантів (вірогідніше він буде найбільш точним).

Приклад підрахунку відсотку вірогідної наявності в зображенні повідомлення англійською мовою (стеганографічних даних) відбувався так: за еталонний порядок відсортованих розрядів додаємо 45%, за еталонну кількість “1” у 7 розряді додавалось 35%, за еталонну різницю між сусідніми розрядами – 19%. Розподіл на відсотки відбувався за пріоритетною значимістю кожного з пунктів. Отже, максимальний відсоток, який можна отримати, це 99%, адже програма не дає 100% гарантії наявності в зображенні стеганографічних даних.

Початковий файл аналізу з растровим зображення BMP-формату з ім’ям "inpic.bmp" має наступний вигляд (рис. 2.20):

Рис. 2.20. Вхідне зображення для аналізу

Система використовує консольний вигляд для виводу результату та має наступний вид (рис. 2.21):

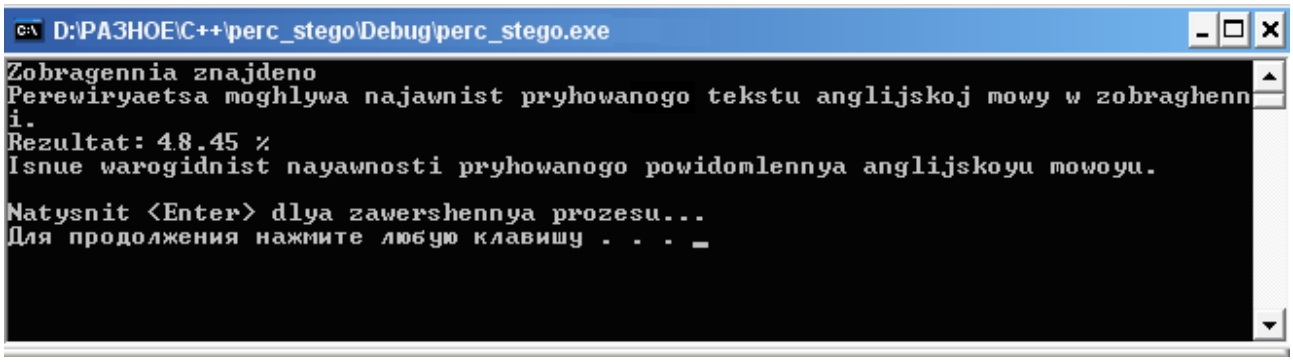


Рис. 2.21. Результат роботи програми

В результаті роботи відображається найбільший відсоток вірогідної наявності з чотирьох результатів розрахунків проведених тестів на виявлення англійського тексту в заданому растровому зображенні з текстовим поясненням цього результату («Результат: 18.45 %, є вірогідність наявності прихованого повідомлення англійською мовою»).

Оцінка результатів виконується таким чином:

– якщо найбільший відсоток вірогідної наявності (per) > 70, то виводиться повідомлення "Дуже велика вірогідність наявності прихованого повідомлення англійською мовою.";

– якщо per > 50 та <69, то "Велика вірогідність наявності прихованого повідомлення англійською мовою.";

– якщо per > 30 та <49, то "Є вірогідність наявності прихованого повідомлення англійською мовою.";

– якщо per >= 0 та <29, то "Дуже мала вірогідність наявності прихованого повідомлення англійською мовою."

При перевірці іншого зображення (рис. 2.21) отриманий відсоток набагато менший (8,5%, що означає про дуже малу вірогідність наявності прихованого повідомлення англійською мовою) (рис. 2.22):

Рис. 2.21. Зображення для аналізу

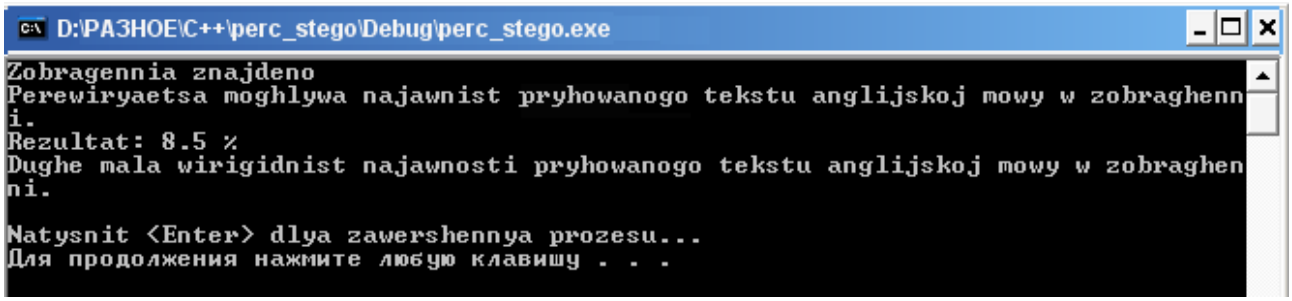


Рис. 2.22. Результат аналізу зображення

При цьому, можна зробити висновок, що в другому зображенні вірогідність наявності прихованого повідомлення англійською мовою значно менша, а ніж в першому, що може свідчити про реальну наявність стенографічні даних в першому зображенні.

Якщо при зчитуванні зображення файл буде відсутній або зображення не в BMP-форматі, то програма видає помилку й пропонує перевірити наявність необхідного файлу(зображення) та перезапустити програму (рис. 2.23).

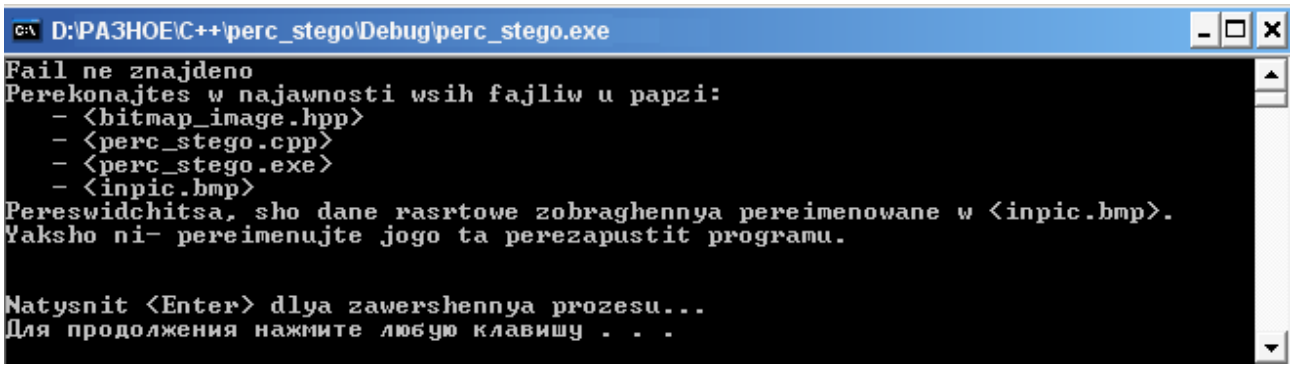


Рис. 2.23. Повідомлення про помилку в роботі системи

2.7.5. Тестування програми

Створений алгоритм було додатково досліджено на криптостійкість та складність.

Для дослідження криптостійкості було побудовано розподіли байт до й після шифрування (рис. 2.24).

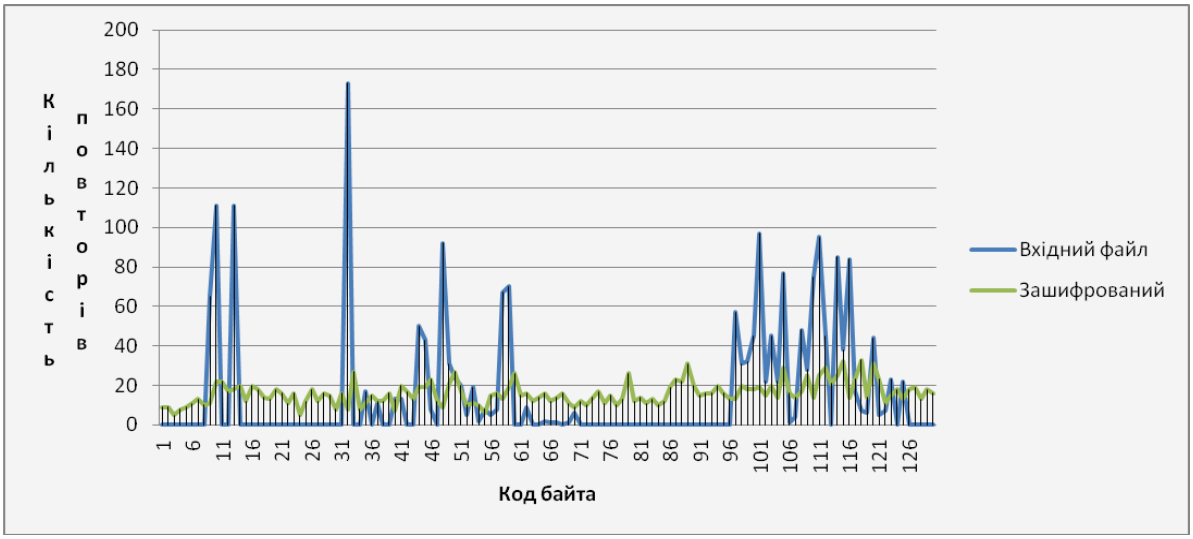


Рис. 2.24. Розподіл значень байтів у зображенні до та після включення текстового повідомлення

Вхідним файлом був англомовний текст (лише стандартні символи ASCII, тобто байти приймали значення від 0 до 127). На розподілі чітко видно піки, що відповідають найбільш часто вживаним символам. Найвищий пік спостерігається в точці 32, що відповідає символу «пробіл» у кодуванні ASCII. Два піки однакової висоти в точках 10 і 13 відповідають переносам рядка (в текстових файлах Microsoft Windows використовується перенос рядка CR+LF). Інші піки відповідають найбільш уживаним символам англійської мови та знакам пунктуації. Після шифрування розподіл байт близький до рівномірного. Це означає, що запропонований у роботі алгоритм потенційно є стійким до атак на основі частотного аналізу. Аналогічний експеримент був проведений для даних різного об'єму й різних типів. У всіх них було одержано розподіли, близькі до рівномірного.

Також створений алгоритм було досліджено на стійкість до диференціальної атаки за допомогою індикаторів NPCR (Number of Pixels Change Rate) та UACI (Unified Average Changing Intensity), та порівняно з результатами тестування алгоритму Zhang Yong, який використовує систему тьюрмітів для шифрування фотографій. Нехай, C1 та C2 – два зашифрованих файла. Тоді індикаторів NPCR та UACI розраховуються так:

$$\mathrm{NPCR} = \frac{1}{\mathrm{MN}} \sum_{i=1}^{M} \sum_{j=1}^{N} (Sign(C_1(i,j) - C_2(i.j))) * 100\%, \tag{2.2}$$

$$\mathrm{UACI} \frac{1}{25\mathrm{sMN}} \sum_{i=1}^{M} \sum_{j=1}^{N} (C_1(i,j) - C_2(i.j)) * 100\% \ , \tag{2.3}$$

.

Ідеальними значеннями для ідентифікаторів є 99.61% для NPCR і 33.46% для UACI.

Значення розрахованих індикаторів NPCR для різних файлів знаходилося в межах від 99.85 до 99.90, а UACI від 24.56 до 40.52, що в окремих випадках було краще за алгоритм, запропонований Zhang Yong. Отже, можна сказати, що даний алгоритм є стійким до диференціальних атак.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів - 1500;
- коефіцієнт складності програми - 2;
- коефіцієнт корекції програми в ході її розробки - 0,08;
- годинна заробітна плата програміста, грн./год - 50.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де to – витрати праці на підготовку й опис поставленої задачі (50),

ти – витрати праці на дослідження алгоритму рішення задачі,

та – витрати праці на розробку блок-схеми алгоритму,

тп – витрати праці на програмування по готовій блок-схемі,

тотл – витрати праці на налагодження програми на ЕОМ,

тд – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \quad (3.2)$$

де q - передбачуване число операторів,

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

$$Q = 1500 \cdot 2 \cdot (1 + 0{,}08) = 3240 \text{ людино-годин.} \quad (3.3)$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K}, \text{ людино-годин,} \quad (3.4)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; B=1.2 … 1.5,

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{3240 \cdot 1{,}3}{80 \cdot 1{,}2} = 44, \text{ людино-годин.} \quad (3.5)$$

Витрати на складання програми по готовій блок-схемі:

$$t_a = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \quad (3.6)$$

$$t_a = \frac{3240}{22 \cdot 1{,}2} = 123 \text{ людино-годин.} \quad (3.7)$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.8}$$

$$t_n = \frac{3240}{23 \cdot 1{,}2} = 117 \text{ людино-годин.} \tag{3.9}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.10}$$

$$t_{отл} = \frac{3240}{5 \cdot 1{,}2} = 540 \text{ людино-годин.} \tag{3.11}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.12}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.13}$$

$$t_{др} = \frac{3240}{18 \cdot 1{,}2} = 150 \text{ людино-годин,} \tag{3.14}$$

де tдо - трудомісткість редагування, печатки й оформлення документації.

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \tag{3.15}$$

$$t_{до} = 150 + 73 = 223 \text{ людино-годин.} \tag{3.16}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 44 + 123 + 117 + 540 + 223 = 1097 \text{ людино-годин.} \tag{3.17}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (3.18)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.19)$$

де t - загальна трудомісткість, людино-годин,

Спр - середня годинна заробітна плата програміста, грн/година.

$$Ç_{Çï} = 1097 \cdot 50 = 54850 \text{ ãðí.} \quad (3.20)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times C_{М}, \text{ грн,} \quad (3.21)$$

де tотл - трудомісткість налагодження програми на ЕОМ, год.,

Смч - вартість машино-години ЕОМ, грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$Ç_{ÎÂ} = 540 \times 10 = 5400 \text{ ãðí.}, \quad (3.22)$$

$$Ê_{ïï} = 54850 + 5400 = 60250 \text{ ãðí.} \quad (3.23)$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.,} \tag{3.24}$$

де Bk - число виконавців,

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{1097}{1 \cdot 176} = 6 \text{ міс.} \tag{3.25}$$

Визначено трудомісткість розробленої інформаційної системи (1097 люд-год), проведений підрахунок вартості роботи по створенню програми (60250 грн.) та розраховано час на його створення (6 міс).

ВИСНОВКИ

Метою кваліфікаційної роботи є створенні додатка, що надає можливість здійснювати аналіз статистичних розподілів бітів у зображеннях, що містять стеганографічні дані, виявляти їх характерні риси, та за допомогою програми виявляти їх в файлах з растровими зображеннях BMP-формату.

Стеганографія може використовуватися зловмисниками для безпечної передачі секретних даних, які в результаті можуть виявитися небезпечними для суспільства, а найпростіше передавати такі дані через зображення.

Методів виявлення наявності стеганографічних даних багато, але існує недостатня кількість програмних реалізацій цих методів. В даній роботі запропоновано алгоритм для виявлення наявності схованого повідомлення англійською мовою в растрових зображеннях BMP-формату.

Проаналізувавши різні методи виявлення стеганографічних даних виявлено, що ринок програмного забезпечення недостатньо забезпечений розробками для розв’язання даної проблеми. Створена інформаційна система “Perc_stego” стане у нагоді і розробникам месенджерів, і державним установам, воно зробить цифровий світ безпечнішим.

Під час роботи над даним проектом визначено, що через велику кількість алгоритмів шифрування та різноманіття контейнерів не існує універсального методу виявлення стеганографічних даних. Тому потрібно орієнтуватися на створення ПЗ, яке б визначало стеганознаки з більшою відсотковою точністю.

Запропонований в роботі алгоритм передбачає, що стеганографічним контейнером є растрове зображення, в якому секретні дані приховані в молодших бітах значень кольорових каналів кожного пікселя. Проведений аналіз розподілів бітів у зображеннях, одержаних за допомогою популярних стеганографічних програм, дозволив виділити характерні ознаки, що можуть свідчити про наявність прихованих даних, що було покладено в основу алгоритму.

Головна цінність розробленої системи полягає в тому, що створене в її

рамках програмне забезпечення може бути використане для виявлення потенційних загроз різної природи. Запропонований алгоритм може бути також адаптований для інших видів стеганографічних контейнерів.

Основним результатом роботи є алгоритм та програмний додаток для виявлення можливої наявності стеганографічних даних у растрових зображеннях. Програмне забезпечення створене за допомогою мови програмування C++, яка має достатній функціонал для реалізації запропонованого алгоритму.

Під час тестування суттєвих недоліків у роботі програми не виявлено. Отже, завдання роботи виконано в повному обсязі.

В економічному розділі були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 6 місяців, трудомісткість розробки ПЗ – 1097 людино-годин, а витрати на її створення програмного забезпечення - 60250 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аграновский А.В., Балакин А.В., Грибунин В.Г., Сапожников С.А. Стеганография, цифровые водяные знаки и стеганоанализ. – М.: Вузовская книга, 2009. – 220 с.

2. Барсуков В.С., Романцов А.П. Оценка уровня скрытности мультимедийных стеганографических каналов хранения и передачи информации // Специальная Техника. – 2000. – № 1

3. Быков С. Ф., Мотуз О. В. Основы стегоанализа // Защита информации. Конфидент. - СПб.: 2000. - № 3. - С. 38-41

4. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

5. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 15.03.2021.

6. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2021.

7. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

8. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

9. Иванов М.А., Чугунков И.В. Теория, применение и оценка качества генераторов псевдослучайных после - довательностей. – М.: КУДИЦ-ОБРАЗ, 2003.

10. Конахович Г.Ф., Пузыренко А.Ю. Компьютерная стеганография. Теория и практика. – Киев: МК-Пресс, 2006. – 288 с.

11. Колобова А.К., Колобов Д.Г., Герасимов А.С. Стеганография от древности до наших дней // Безопасность информационных технологий. — Москва: КлАССное снаряжение, 2015. - № 4. - С. 71-74.

12. Коркач І.В., Пирогова Ю.І. Використання технологій IP-телефонії для прихованої передачі інформації.. — Інформаційна безпека людини, суспільства, держави, 2012. - Vol. 9, no. 2. - P. 124–128. - DOI:004.056.5 (045).

13. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

14. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 121 «Програмна інженерія» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

15. Пузиренко, О. Ю.. Система обробки додаткової інформації в мережі цифрового звукового мовлення. Проблеми інформатизації та управління (uk) 2 (34).-2011. С. 111–119. ISSN 2073-4751. Процитовано 2017-05-16.

16. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

17. Стеганографія : навч. посіб (плагіат (реалізація методів, додатки) без посилань на монографії Грібуніна, Окова, Турінцева, Конаховича, Пузиренко). / О. О. Кузнецов, С. П. Євсеєв,О. Г. Король ; М-во освіти і науки, молоді та

спорту України, Харк. нац. екон. ун-т. – Х. : Вид-во ХНЕУ, 2011. – 232 с. : іл. – Бібліогр.: с. 202-206 (67 назв). – ISBN 978-966-676-483-9

18. Сушко С.А. Лекция 9. Практическая криптология – URL: http://bit.nmu.org.ua/ua дата звернення: 5.04.2019.

19. Хорошко В., Чекатков А. Методы и средства защиты информации. – К.: Юниор, 2003. – 504 с.

20. Юдін О. К. Удосконалення стеганографічних методів на базі аналізу колірних моделей зображення / О. К. Юдін, Я. А. Симониченко // Наукоємні технології. – 2012. – №1 (13). – С. 70-75.

ДОДАТОК А

КОД ПРОГРАМИ

perc_stego.cpp

```
#include <bits/stdc++.h>
#include "bitmap_image.hpp"
using namespace std;

int bit_LSB(string fis,string sec,string thi,string fou)
{
   char k[100];
   int a[8];
   k[0] = fis[0];
   k[1] = fis[1];
   k[2] = sec[0];
   k[3] = sec[1];
   k[4] = thi[0];
   k[5] = thi[1];
   k[6] = fou[0];
   k[7] = fou[1];
   a[0] = static_cast<int>(k[0])-48;
   a[1] = static_cast<int>(k[1])-48;
   a[2] = static_cast<int>(k[2])-48;
   a[3] = static_cast<int>(k[3])-48;
   a[4] = static_cast<int>(k[4])-48;
   a[5] = static_cast<int>(k[5])-48;
   a[6] = static_cast<int>(k[6])-48;
   a[7] = static_cast<int>(k[7])-48;
   return
pow(2,7)*a[0]+pow(2,6)*a[1]+pow(2,5)*a[2]+pow(2,4)*a[3]+pow(2,3)*a[4]+pow(2,2)*a[5]+pow(
2,1)*a[6]+a[7];
}

string least_bit(unsigned char k)
```

```
{
    const int i = (int)k;
    const int second = i%2;
    const int first = (i/2)%2;
    char bit[1];
    bit[0] = first+'0';
    bit[1] = second+'0';
    return bit;
}

int rozriad(int stepen,int a)
{
    double n = pow(2,stepen);
    int b = (int) n;
    return (a/b)%2;
}

int odds_rozr(long int rozr_count[])
{
    int odds_zagal = 0;
    if ((rozr_count[5]-rozr_count[6]) >= 0)
        odds_zagal += (rozr_count[5]-rozr_count[6]);
    if ((rozr_count[6]-rozr_count[2]) >= 0)
        odds_zagal += (rozr_count[6]-rozr_count[2]);
    if ((rozr_count[2]-rozr_count[0]) >= 0)
        odds_zagal += (rozr_count[2]-rozr_count[0]);
    if ((rozr_count[0]-rozr_count[1]) >= 0)
        odds_zagal += (rozr_count[0]-rozr_count[1]);
    if ((rozr_count[1]-rozr_count[3]) >= 0)
        odds_zagal += (rozr_count[1]-rozr_count[3]);
    if ((rozr_count[3]-rozr_count[4]) >= 0)
        odds_zagal += (rozr_count[3]-rozr_count[4]);
    if ((rozr_count[4]-rozr_count[7]) >= 0)
        odds_zagal += (rozr_count[4]-rozr_count[7]);
    return odds_zagal;
```

```
}

int main()
{
  setlocale(LC_ALL, "Ukrainian");
  bitmap_image image("inpic.bmp");
  if (!image)
  {
    cout<<"Не знайдено"<<endl;
    cout<<"Переконайтесь, що у пацпі знаходиться всі необхідні файли:"<<endl;
    cout<<"   - <bitmap_image.hpp>"<<endl;
    cout<<"   - <perc_stego.cpp>"<<endl;
    cout<<"   - <perc_stego.exe>"<<endl;
    cout<<"   - <inpic.bmp>"<<endl;
    cout<<"Пересвідчитеся, що дане растрове зображення перейменоване у
<inpic.bmp>."<<endl;
    cout<<"Якщо - ні, то перейменуйте зображення та перезапустіть програму."<<endl;
    cout<<endl;
    cout<<"Натисніть <Enter> для завершення процесу..."<<endl;
    system("pause");
    return 1;
  }
{
   const int i = (int)k;
   const int second = i%2;
   const int first = (i/2)%2;
   char bit[1];
   bit[0] = first+'0';
   bit[1] = second+'0';
   return bit;
}

int rozriad(int stepen,int a)
{
```

```
    double n = pow(2,stepen);
    int b = (int) n;
    return (a/b)%2;
}

int odds_rozr(long int rozr_count[])
{
    int odds_zagal = 0;
    if ((rozr_count[5]-rozr_count[6]) >= 0)
        odds_zagal += (rozr_count[5]-rozr_count[6]);
    if ((rozr_count[6]-rozr_count[2]) >= 0)
        odds_zagal += (rozr_count[6]-rozr_count[2]);
    if ((rozr_count[2]-rozr_count[0]) >= 0)
        odds_zagal += (rozr_count[2]-rozr_count[0]);
    if ((rozr_count[0]-rozr_count[1]) >= 0)
        odds_zagal += (rozr_count[0]-rozr_count[1]);
    if ((rozr_count[1]-rozr_count[3]) >= 0)
        odds_zagal += (rozr_count[1]-rozr_count[3]);
    if ((rozr_count[3]-rozr_count[4]) >= 0)
        odds_zagal += (rozr_count[3]-rozr_count[4]);
    if ((rozr_count[4]-rozr_count[7]) >= 0)
        odds_zagal += (rozr_count[4]-rozr_count[7]);
    return odds_zagal;

}

int main()
{
  setlocale(LC_ALL, "Ukrainian");
  bitmap_image image("inpic.bmp");
  if (!image)
  {
    cout<<"Не знайдено"<<endl;
    cout<<"Переконайтесь, що у пацпі знаходиться всі необхідні файли:"<<endl;
    cout<<"   - <bitmap_image.hpp>"<<endl;
```

```
    cout<<"   - <perc_stego.cpp>"<<endl;
    cout<<"   - <perc_stego.exe>"<<endl;
    cout<<"   - <inpic.bmp>"<<endl;
    cout<<"Пересвідчитеся, що дане растрове зображення перейменоване у
<inpic.bmp>."<<endl;
    cout<<"Якщо - ні, то перейменуйте зображення та перезапустіть програму."<<endl;
    cout<<endl;
    cout<<"Натисніть <Enter> для завершення процесу..."<<endl;
    system("pause");
    return 1;
  }
  cout<<"Зображення знайдено"<<endl;
  cout<<"Перевіряється можлива наявність прихованого тексту англійської мови в
зображенні."<<endl;
  cout<<endl;
  long int numi = 0;
  int num_bite;
  long int rozr_count[8] = {0};
  double ammount_bite = 0;
  string main_number[5];
  const unsigned int height = image.height();
  const unsigned int width  = image.width();

  //Тест 1

  for (size_t y = 0; y < height; ++y)
  {
    for (size_t x = 0; x < width; ++x)
    {
      rgb_t colour;
      image.get_pixel(x, y, colour);
      numi++;
      main_number[numi] = least_bit(colour.blue);
      if (numi == 4)
      {
```

```
            num_bite = bit_LSB(main_number[1],main_number[2],main_number[3],main_number[4]);
            ammount_bite += 1;
            rozr_count[0] += num_bite%2;
            rozr_count[1] += rozriad(1,num_bite);
            rozr_count[2] += rozriad(2,num_bite);
            rozr_count[3] += rozriad(3,num_bite);
            rozr_count[4] += rozriad(4,num_bite);
            rozr_count[5] += rozriad(5,num_bite);
            rozr_count[6] += rozriad(6,num_bite);
            rozr_count[7] += rozriad(7,num_bite);
            numi = 0;
        }
      }

   cout<<"Зображення знайдено"<<endl;
   cout<<"Перевіряється можлива наявність прихованого тексту англійської мови в
зображенні."<<endl;
   cout<<endl;
   long int numi = 0;
   int num_bite;
   long int rozr_count[8] = {0};
   double ammount_bite = 0;
   string main_number[5];
   const unsigned int height = image.height();
   const unsigned int width  = image.width();

   //Тест 1

   for (size_t y = 0; y < height; ++y)
   {
     for (size_t x = 0; x < width; ++x)
     {
       rgb_t colour;
       image.get_pixel(x, y, colour);
       numi++;
```

```
        main_number[numi] = least_bit(colour.blue);
        if (numi == 4)
        {
           num_bite = bit_LSB(main_number[1],main_number[2],main_number[3],main_number[4]);
           ammount_bite += 1;
           rozr_count[0] += num_bite%2;
           rozr_count[1] += rozriad(1,num_bite);
           rozr_count[2] += rozriad(2,num_bite);
           rozr_count[3] += rozriad(3,num_bite);
           rozr_count[4] += rozriad(4,num_bite);
           rozr_count[5] += rozriad(5,num_bite);
           rozr_count[6] += rozriad(6,num_bite);
           rozr_count[7] += rozriad(7,num_bite);
           numi = 0;
        }
      }
    }
{
   const int i = (int)k;
   const int second = i%2;
   const int first = (i/2)%2;
   char bit[1];
   bit[0] = first+'0';
   bit[1] = second+'0';
   return bit;
}

int rozriad(int stepen,int a)
{
   double n = pow(2,stepen);
   int b = (int) n;
   return (a/b)%2;
}

int odds_rozr(long int rozr_count[])
```

```
{
   int odds_zagal = 0;
   if ((rozr_count[5]-rozr_count[6]) >= 0)
      odds_zagal += (rozr_count[5]-rozr_count[6]);
   if ((rozr_count[6]-rozr_count[2]) >= 0)
      odds_zagal += (rozr_count[6]-rozr_count[2]);
   if ((rozr_count[2]-rozr_count[0]) >= 0)
      odds_zagal += (rozr_count[2]-rozr_count[0]);
   if ((rozr_count[0]-rozr_count[1]) >= 0)
      odds_zagal += (rozr_count[0]-rozr_count[1]);
   if ((rozr_count[1]-rozr_count[3]) >= 0)
      odds_zagal += (rozr_count[1]-rozr_count[3]);
   if ((rozr_count[3]-rozr_count[4]) >= 0)
      odds_zagal += (rozr_count[3]-rozr_count[4]);
   if ((rozr_count[4]-rozr_count[7]) >= 0)
      odds_zagal += (rozr_count[4]-rozr_count[7]);
   return odds_zagal;

}

int main()
{
  setlocale(LC_ALL, "Ukrainian");
  bitmap_image image("inpic.bmp");
  if (!image)
  {
    cout<<"Не знайдено"<<endl;
    cout<<"Переконайтесь, що у пацпі знаходиться всі необхідні файли:"<<endl;
    cout<<"   - <bitmap_image.hpp>"<<endl;
    cout<<"   - <perc_stego.cpp>"<<endl;
    cout<<"   - <perc_stego.exe>"<<endl;
    cout<<"   - <inpic.bmp>"<<endl;
    cout<<"Пересвідчитеся, що дане растрове зображення перейменоване у
<inpic.bmp>."<<endl;
    cout<<"Якщо - ні, то перейменуйте зображення та перезапустіть програму."<<endl;
```

```
    cout<<endl;
    cout<<"Натисніть <Enter> для завершення процесу..."<<endl;
    system("pause");
    return 1;
  }
  cout<<"Зображення знайдено"<<endl;
  cout<<"Перевіряється можлива наявність прихованого тексту англійської мови в
зображенні."<<endl;
  cout<<endl;
  long int numi = 0;
  int num_bite;
  long int rozr_count[8] = {0};
  double ammount_bite = 0;
  string main_number[5];
  const unsigned int height = image.height();
  const unsigned int width  = image.width();

  //Тест 1

  for (size_t y = 0; y < height; ++y)
  {
    for (size_t x = 0; x < width; ++x)
    {
      rgb_t colour;
      image.get_pixel(x, y, colour);
      numi++;
      main_number[numi] = least_bit(colour.blue);
      if (numi == 4)
      {
        num_bite = bit_LSB(main_number[1],main_number[2],main_number[3],main_number[4]);
        ammount_bite += 1;
        rozr_count[0] += num_bite%2;
        rozr_count[1] += rozriad(1,num_bite);
        rozr_count[2] += rozriad(2,num_bite);
        rozr_count[3] += rozriad(3,num_bite);
```

```
        rozr_count[4] += rozriad(4,num_bite);
        rozr_count[5] += rozriad(5,num_bite);
        rozr_count[6] += rozriad(6,num_bite);
        rozr_count[7] += rozriad(7,num_bite);
        numi = 0;
    }
  }

double percentage_1 = 0;
double am_7 = (double) rozr_count[7];
if (((am_7/ammount_bite)*35) <= 35)
   percentage_1 += 35-((am_7/ammount_bite)*35);
double koef = (double) odds_rozr(rozr_count);
double etalon = 7*ammount_bite*0.05;
double z;
if (etalon > koef)
{
   z = 19-((etalon-koef)/etalon)*19;
}
else
{
   z = 19-((koef-etalon)/etalon)*19;
   if ((z > 19) || (z < 0)) z = 0;
}
percentage_1 += z;
int rozr_bit[8];
for (int i = 0; i <= 7; i++) rozr_bit[i]=i;
for(int i = 0; i <= 7; i++)
{
   for (int j = 7; j > i; j--)
   {
     if (rozr_count[j-1] < rozr_count[j])
     {
        swap(rozr_count[j],rozr_count[j-1]);
        swap(rozr_bit[j],rozr_bit[j-1]);
```

```
        }
      }
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
      percentage_1 += 45;
    }
    if ((rozr_bit[0] == 6) && (rozr_bit[1] == 5) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
      percentage_1 += 30;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 2) && (rozr_bit[2] == 6) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
      percentage_1 += 23;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 0) && (rozr_bit[3] == 2) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
      percentage_1 += 20;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 1) &&
(rozr_bit[4] == 0) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
      percentage_1 += 17;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 3) && (rozr_bit[5] == 1) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
      percentage_1 += 21;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 4) && (rozr_bit[6] == 3) && (rozr_bit[7] == 7))
```

```
    {
        percentage_1 += 25;
    }

    //Тест 2
{
    const int i = (int)k;
    const int second = i%2;
    const int first = (i/2)%2;
    char bit[1];
    bit[0] = first+'0';
    bit[1] = second+'0';
    return bit;
}

int rozriad(int stepen,int a)
{
    double n = pow(2,stepen);
    int b = (int) n;
    return (a/b)%2;
}

int odds_rozr(long int rozr_count[])
{
    int odds_zagal = 0;
    if ((rozr_count[5]-rozr_count[6]) >= 0)
        odds_zagal += (rozr_count[5]-rozr_count[6]);
    if ((rozr_count[6]-rozr_count[2]) >= 0)
        odds_zagal += (rozr_count[6]-rozr_count[2]);
    if ((rozr_count[2]-rozr_count[0]) >= 0)
        odds_zagal += (rozr_count[2]-rozr_count[0]);
    if ((rozr_count[0]-rozr_count[1]) >= 0)
        odds_zagal += (rozr_count[0]-rozr_count[1]);
    if ((rozr_count[1]-rozr_count[3]) >= 0)
        odds_zagal += (rozr_count[1]-rozr_count[3]);
```

```
    if ((rozr_count[3]-rozr_count[4]) >= 0)
        odds_zagal += (rozr_count[3]-rozr_count[4]);
    if ((rozr_count[4]-rozr_count[7]) >= 0)
        odds_zagal += (rozr_count[4]-rozr_count[7]);
    return odds_zagal;
{
    int odds_zagal = 0;
    if ((rozr_count[5]-rozr_count[6]) >= 0)
        odds_zagal += (rozr_count[5]-rozr_count[6]);
    if ((rozr_count[6]-rozr_count[2]) >= 0)
        odds_zagal += (rozr_count[6]-rozr_count[2]);
    if ((rozr_count[2]-rozr_count[0]) >= 0)
        odds_zagal += (rozr_count[2]-rozr_count[0]);
    if ((rozr_count[0]-rozr_count[1]) >= 0)
        odds_zagal += (rozr_count[0]-rozr_count[1]);
    if ((rozr_count[1]-rozr_count[3]) >= 0)
        odds_zagal += (rozr_count[1]-rozr_count[3]);
    if ((rozr_count[3]-rozr_count[4]) >= 0)
        odds_zagal += (rozr_count[3]-rozr_count[4]);
    if ((rozr_count[4]-rozr_count[7]) >= 0)
        odds_zagal += (rozr_count[4]-rozr_count[7]);
    return odds_zagal;
}
}

int main()
{
  setlocale(LC_ALL, "Ukrainian");
  bitmap_image image("inpic.bmp");
  if (!image)
  {
    cout<<"Не знайдено"<<endl;
    cout<<"Переконайтесь, що у пацпі знаходиться всі необхідні файли:"<<endl;
    cout<<"   - <bitmap_image.hpp>"<<endl;
    cout<<"   - <perc_stego.cpp>"<<endl;
```

```
    cout<<"   - <perc_stego.exe>"<<endl;
    cout<<"   - <inpic.bmp>"<<endl;
    cout<<"Пересвідчитеся, що дане растрове зображення перейменоване у
<inpic.bmp>."<<endl;
    cout<<"Якщо - ні, то перейменуйте зображення та перезапустіть програму."<<endl;
    cout<<endl;
    cout<<"Натисніть <Enter> для завершення процесу..."<<endl;
    system("pause");
    return 1;
  }
  cout<<"Зоображення знайдено"<<endl;
  cout<<"Перевіряється можлива наявність прихованого тексту англійської мови в
зображенні."<<endl;
  cout<<endl;
  long int numi = 0;
  int num_bite;
  long int rozr_count[8] = {0};
  double ammount_bite = 0;
  string main_number[5];
  const unsigned int height = image.height();
  const unsigned int width  = image.width();

  //Тест 1

  for (size_t y = 0; y < height; ++y)
  {
    for (size_t x = 0; x < width; ++x)
    {
      rgb_t colour;
      image.get_pixel(x, y, colour);
      numi++;
      main_number[numi] = least_bit(colour.blue);
      if (numi == 4)
      {
         num_bite = bit_LSB(main_number[1],main_number[2],main_number[3],main_number[4]);
```

```
        ammount_bite += 1;
        rozr_count[0] += num_bite%2;
        rozr_count[1] += rozriad(1,num_bite);
        rozr_count[2] += rozriad(2,num_bite);
        rozr_count[3] += rozriad(3,num_bite);
        rozr_count[4] += rozriad(4,num_bite);
        rozr_count[5] += rozriad(5,num_bite);
        rozr_count[6] += rozriad(6,num_bite);
        rozr_count[7] += rozriad(7,num_bite);
        numi = 0;
      }
    }

num_bite = 0;
ammount_bite = 0;
for (int i = 0; i <= 7; i++) rozr_count[i] = 0;
numi = 0;
for (size_t y = 0; y < height; ++y)
{
  for (size_t x = 0; x < width; ++x)
  {
    rgb_t colour;
    image.get_pixel(x, y, colour);
    numi++;
    main_number[numi] = least_bit(colour.red);
    if (numi == 4)
    {
      num_bite = bit_LSB(main_number[1],main_number[2],main_number[3],main_number[4]);
      ammount_bite += 1;
      rozr_count[0] += num_bite%2;
      rozr_count[1] += rozriad(1,num_bite);
      rozr_count[2] += rozriad(2,num_bite);
      rozr_count[3] += rozriad(3,num_bite);
      rozr_count[4] += rozriad(4,num_bite);
      rozr_count[5] += rozriad(5,num_bite);
```

```
            rozr_count[6] += rozriad(6,num_bite);
            rozr_count[7] += rozriad(7,num_bite);
            numi = 0;
        }
      }
{
    int odds_zagal = 0;
    if ((rozr_count[5]-rozr_count[6]) >= 0)
        odds_zagal += (rozr_count[5]-rozr_count[6]);
    if ((rozr_count[6]-rozr_count[2]) >= 0)
        odds_zagal += (rozr_count[6]-rozr_count[2]);
    if ((rozr_count[2]-rozr_count[0]) >= 0)
        odds_zagal += (rozr_count[2]-rozr_count[0]);
    if ((rozr_count[0]-rozr_count[1]) >= 0)
        odds_zagal += (rozr_count[0]-rozr_count[1]);
    if ((rozr_count[1]-rozr_count[3]) >= 0)
        odds_zagal += (rozr_count[1]-rozr_count[3]);
    if ((rozr_count[3]-rozr_count[4]) >= 0)
        odds_zagal += (rozr_count[3]-rozr_count[4]);
    if ((rozr_count[4]-rozr_count[7]) >= 0)
        odds_zagal += (rozr_count[4]-rozr_count[7]);
    return odds_zagal;

}

   }
    double percentage_2 = 0;
    am_7 = (double) rozr_count[7];
    if (((am_7/ammount_bite)*35) <= 35)
        percentage_2 += 35-((am_7/ammount_bite)*35);
    koef = (double) odds_rozr(rozr_count);
    etalon = 7*ammount_bite*0.05;
    z = 0;
    if (etalon > koef)
    {
```

```
    z = 19-((etalon-koef)/etalon)*19;
  }
  else
  {
    z = 19-((koef-etalon)/etalon)*19;
    if ((z > 19) || (z < 0)) z = 0;
  }
  percentage_2 += z;
  for (int i = 0; i <= 7; i++) rozr_bit[i]=i;
  for(int i = 0; i <= 7; i++)
  {
    for (int j = 7; j > i; j--)
    {
      if (rozr_count[j-1] < rozr_count[j])
      {
        swap(rozr_count[j],rozr_count[j-1]);
        swap(rozr_bit[j],rozr_bit[j-1]);
      }
    }
  }
  if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
  {
    percentage_2 += 45;
  }
  if ((rozr_bit[0] == 6) && (rozr_bit[1] == 5) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
  {
    percentage_2 += 30;
  }
  if ((rozr_bit[0] == 5) && (rozr_bit[1] == 2) && (rozr_bit[2] == 6) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
  {
    percentage_2 += 23;
  }
```

```
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 0) && (rozr_bit[3] == 2) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
       percentage_2 += 20;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 1) &&
(rozr_bit[4] == 0) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
       percentage_2 += 17;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 3) && (rozr_bit[5] == 1) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
       percentage_2 += 21;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 4) && (rozr_bit[6] == 3) && (rozr_bit[7] == 7))
    {
       percentage_2 += 25;
    }

    //Тест 3

   num_bite = 0;
   ammount_bite = 0;
   for (int i = 0; i <= 7; i++) rozr_count[i] = 0;
   numi = 0;
   for (size_t y = 0; y < height; ++y)
   {
     for (size_t x = 0; x < width; ++x)
     {
       rgb_t colour;
       image.get_pixel(x, y, colour);
       numi++;
       main_number[numi] = least_bit(colour.green);
```

```
    if (numi == 4)
    {
       num_bite = bit_LSB(main_number[1],main_number[2],main_number[3],main_number[4]);
       ammount_bite += 1;
       rozr_count[0] += num_bite%2;
       rozr_count[1] += rozriad(1,num_bite);
       rozr_count[2] += rozriad(2,num_bite);
       rozr_count[3] += rozriad(3,num_bite);
       rozr_count[4] += rozriad(4,num_bite);
       rozr_count[5] += rozriad(5,num_bite);
       rozr_count[6] += rozriad(6,num_bite);
       rozr_count[7] += rozriad(7,num_bite);
       numi = 0;
    }
  }
}
double percentage_3 = 0;
am_7 = (double) rozr_count[7];
if (((am_7/ammount_bite)*35) <= 35)
   percentage_3 += 35-((am_7/ammount_bite)*35);
koef = (double) odds_rozr(rozr_count);
etalon = 7*ammount_bite*0.05;
z = 0;
if (etalon > koef)
{
   z = 19-((etalon-koef)/etalon)*19;
}
else
{
   z = 19-((koef-etalon)/etalon)*19;
   if ((z > 19) || (z < 0)) z = 0;
}
percentage_3 += z;
for (int i = 0; i <= 7; i++) rozr_bit[i]=i;
for(int i = 0; i <= 7; i++)
```

```
    {
        for (int j = 7; j > i; j--)
        {
          if (rozr_count[j-1] < rozr_count[j])
          {
             swap(rozr_count[j],rozr_count[j-1]);
             swap(rozr_bit[j],rozr_bit[j-1]);
          }
        }
    }
{
    int odds_zagal = 0;
    if ((rozr_count[5]-rozr_count[6]) >= 0)
        odds_zagal += (rozr_count[5]-rozr_count[6]);
    if ((rozr_count[6]-rozr_count[2]) >= 0)
        odds_zagal += (rozr_count[6]-rozr_count[2]);
    if ((rozr_count[2]-rozr_count[0]) >= 0)
        odds_zagal += (rozr_count[2]-rozr_count[0]);
    if ((rozr_count[0]-rozr_count[1]) >= 0)
        odds_zagal += (rozr_count[0]-rozr_count[1]);
    if ((rozr_count[1]-rozr_count[3]) >= 0)
        odds_zagal += (rozr_count[1]-rozr_count[3]);
    if ((rozr_count[3]-rozr_count[4]) >= 0)
        odds_zagal += (rozr_count[3]-rozr_count[4]);
    if ((rozr_count[4]-rozr_count[7]) >= 0)
        odds_zagal += (rozr_count[4]-rozr_count[7]);
    return odds_zagal;
}
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
        percentage_3 += 45;
    }
    if ((rozr_bit[0] == 6) && (rozr_bit[1] == 5) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
```

```
    {
        percentage_3 += 30;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 2) && (rozr_bit[2] == 6) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
        percentage_3 += 23;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 0) && (rozr_bit[3] == 2) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
        percentage_3 += 20;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 1) &&
(rozr_bit[4] == 0) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
        percentage_3 += 17;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 3) && (rozr_bit[5] == 1) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
    {
        percentage_3 += 21;
    }
    if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 4) && (rozr_bit[6] == 3) && (rozr_bit[7] == 7))
    {
        percentage_3 += 25;
    }

    //Тест 4

   num_bite = 0;
   ammount_bite = 0;
   for (int i = 0; i <= 7; i++) rozr_count[i] = 0;
   numi = 0;
```

```
for (size_t y = 0; y < height; ++y)
{
  for (size_t x = 0; x < width; ++x)
  {
    rgb_t colour;
    image.get_pixel(x, y, colour);
    numi++;
    main_number[numi] = least_bit(colour.red);
    if (numi == 4)
    {
       num_bite = bit_LSB(main_number[1],main_number[2],main_number[3],main_number[4]);
       ammount_bite += 1;
       rozr_count[0] += num_bite%2;
       rozr_count[1] += rozriad(1,num_bite);
       rozr_count[2] += rozriad(2,num_bite);
       rozr_count[3] += rozriad(3,num_bite);
       rozr_count[4] += rozriad(4,num_bite);
       rozr_count[5] += rozriad(5,num_bite);
       rozr_count[6] += rozriad(6,num_bite);
       rozr_count[7] += rozriad(7,num_bite);
       numi = 0;
    }
    numi++;
    main_number[numi] = least_bit(colour.green);
    if (numi == 4)
    {
       num_bite = bit_LSB(main_number[1],main_number[2],main_number[3],main_number[4]);
       ammount_bite += 1;
       rozr_count[0] += num_bite%2;
       rozr_count[1] += rozriad(1,num_bite);
       rozr_count[2] += rozriad(2,num_bite);
       rozr_count[3] += rozriad(3,num_bite);
       rozr_count[4] += rozriad(4,num_bite);
       rozr_count[5] += rozriad(5,num_bite);
       rozr_count[6] += rozriad(6,num_bite);
```

```
            rozr_count[7] += rozriad(7,num_bite);
            numi = 0;
        }
        numi++;
        main_number[numi] = least_bit(colour.blue);
        if (numi == 4)
        {
            num_bite = bit_LSB(main_number[1],main_number[2],main_number[3],main_number[4]);
            ammount_bite += 1;
            rozr_count[0] += num_bite%2;
            rozr_count[1] += rozriad(1,num_bite);
            rozr_count[2] += rozriad(2,num_bite);
            rozr_count[3] += rozriad(3,num_bite);
            rozr_count[4] += rozriad(4,num_bite);
            rozr_count[5] += rozriad(5,num_bite);
            rozr_count[6] += rozriad(6,num_bite);
            rozr_count[7] += rozriad(7,num_bite);
            numi = 0;
        }
      }
    }
    double percentage_4 = 0;
    am_7 = (double) rozr_count[7];
    if (((am_7/ammount_bite)*35) <= 35)
       percentage_4 += 35-((am_7/ammount_bite)*35);
    koef = (double) odds_rozr(rozr_count);
    etalon = 7*ammount_bite*0.05;
    z = 0;
    if (etalon > koef)
    {
       z = 19-((etalon-koef)/etalon)*19;
    }
    else
    {
       z = 19-((koef-etalon)/etalon)*19;
```

```
    if ((z > 19) || (z < 0)) z = 0;
  }
  percentage_4 += z;
  for (int i = 0; i <= 7; i++) rozr_bit[i]=i;
  for(int i = 0; i <= 7; i++)
  {
    for (int j = 7; j > i; j--)
    {
      if (rozr_count[j-1] < rozr_count[j])
      {
        swap(rozr_count[j],rozr_count[j-1]);
        swap(rozr_bit[j],rozr_bit[j-1]);
      }
    }
  }
  if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
  {
    percentage_4 += 45;
  }
  if ((rozr_bit[0] == 6) && (rozr_bit[1] == 5) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
  {
    percentage_4 += 30;
  }
  if ((rozr_bit[0] == 5) && (rozr_bit[1] == 2) && (rozr_bit[2] == 6) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
  {
    percentage_4 += 23;
  }
  if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 0) && (rozr_bit[3] == 2) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
  {
    percentage_4 += 20;
  }
```

```
   if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 1) &&
(rozr_bit[4] == 0) && (rozr_bit[5] == 3) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
   {
      percentage_4 += 17;
   }
   if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 3) && (rozr_bit[5] == 1) && (rozr_bit[6] == 4) && (rozr_bit[7] == 7))
   {
      percentage_4 += 21;
   }
   if ((rozr_bit[0] == 5) && (rozr_bit[1] == 6) && (rozr_bit[2] == 2) && (rozr_bit[3] == 0) &&
(rozr_bit[4] == 1) && (rozr_bit[5] == 4) && (rozr_bit[6] == 3) && (rozr_bit[7] == 7))
   {
      percentage_4 += 25;
   }
{
   int odds_zagal = 0;
   if ((rozr_count[5]-rozr_count[6]) >= 0)
      odds_zagal += (rozr_count[5]-rozr_count[6]);
   if ((rozr_count[6]-rozr_count[2]) >= 0)
      odds_zagal += (rozr_count[6]-rozr_count[2]);
   if ((rozr_count[2]-rozr_count[0]) >= 0)
      odds_zagal += (rozr_count[2]-rozr_count[0]);
   if ((rozr_count[0]-rozr_count[1]) >= 0)
      odds_zagal += (rozr_count[0]-rozr_count[1]);
   if ((rozr_count[1]-rozr_count[3]) >= 0)
      odds_zagal += (rozr_count[1]-rozr_count[3]);
   if ((rozr_count[3]-rozr_count[4]) >= 0)
      odds_zagal += (rozr_count[3]-rozr_count[4]);
   if ((rozr_count[4]-rozr_count[7]) >= 0)
      odds_zagal += (rozr_count[4]-rozr_count[7]);
   return odds_zagal;
}
```

```
    double perc = max(max(percentage_1,percentage_2),max(percentage_3,percentage_4));
    cout<<fixed;
    cout.precision(2);
    cout<<"Результат: "<<perc<<"%"<<endl;
    if (perc > 70) cout<<"Дуже велика вірогідність наявності прихованого повідомлення англійською мовою."<<endl;
        else if (perc > 50) cout<<"Велика вірогідність наявності прихованого повідомлення англійською мовою."<<endl;
        else if (perc > 30) cout<<"Є вірогідність наявності прихованого повідомлення англійською мовою."<<endl;
        else if (perc >= 0) cout<<"Дуже мала вірогідність наявності прихованого повідомлення англійською мовою."<<endl;
    cout<<endl;
    cout<<"Натисніть <Enter> для завершення процесу..."<<endl;
    system("pause");
    return 0;
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи