**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУ8

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Басараб М.А.

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине Технологии и методы программирования

Студент группы ИУ8-33

Темиров Тимур Расулович

(Фамилия, имя, отчество)

Тема курсового проекта Реализация алгоритма стеганографии в цифровых изображениях

Направленность КП (учебный, исследовательский, практический, производственный, др.)

учебный

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к 3 нед., 50% к 9 нед., 75% к 12 нед., 100% к 15 нед.

***Задание*** Разработка программы на с++, реализующей алгоритмы стеганографии в изображениях

***Оформление курсового проекта:***

Расчетно-пояснительная записка на \_\_\_\_\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_\_\_ » сентября 2024 г.

**Руководитель курсового проекта**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Колесников

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Т. Р. Темиров

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Реферат**

Записка 21с., 14рис., 0табл., 11источников, 0приложений.

Ключевые слова: безопасность данных, стеганография, цифровые изображения, встраивание в заголовок, LSB.

Цель данной работы состоит в изучении методов стеганографии в цифровых изображениях и разработке алгоритмов для реализации этих методов. Для реализации цели была проведена оценка существующих методов стеганографии, выбраны наиболее популярные среди них и разработана программа на языке с++. Программа протестирована на корректность встраивания в изображение текста.

В результате было создано консольное приложение, которое может:

* Вставить текст в изображение двумя методами: встраивание в заголовок и LSB.
* Прочитать скрытый в изображении текст.
* Сравнить два файла для того, чтобы узнать находится ли в изображении текст.

Программа прошла тесты на встраивание и считывание текста, а значит она пригодна для использования.

У программы несколько областей применения. Во-первых, это скрытие факта наличия информации. Человек может спрятать важную информацию в не привлекающих к себе внимания изображениях. Это бывает полезно, когда человек не хочет, чтобы его прослушивали. Во-вторых, это защита авторских прав. При помощи стеганографии и цифровых водяных знаков можно подтвердить авторское право и пресечь изготовление пиратских копий. Например, автор может встроить в изображение дату и место съёмки.

**Содержание**

Введение ……………………………………………………………………………4

1. Анализ требований и уточнение спецификаций………………………………6

1.1. Метод встраивания в заголовок ……………………………………….7

1.2. Метод сравнивания файлов …………………………………………..10

1.3. Метод LSB (least significant bit) ………………………………………11

2. Проектирование структуры и компонентов программного продукта ..….…13

3. Выбор стратегии тестирования и разработка тестов ..……………………….17

Заключение ………………………………………………………………………..20

Литература ………………………………………………………………………...21

**Введение**

В XXI веке постоянно совершенствуется компьютерная техника. С каждым годом цифровизация набирает новые обороты. Из-за этого актуальность защиты информации растёт, так как информация - самый ценный ресурс в современном мире. Люди хранят свои паспортные данные, деньги, увлечения и хобби в Интернете. Злоумышленники могут использовать любую информацию против человека, поэтому безопасность в интернете очень важна.

Стеганография — это наука о скрытой передаче информации, путём сохранения в тайне самого факта передачи. Она появилась задолго до криптографии, потому что ещё в древние времена, до появления первых шифров, люди хотели защитить информацию от несанкционированного доступа. Одним из самых популярных примеров являются симпатические чернила. Эти чернила становятся видимыми только при определённых условиях, например, нагрев, освещение. Даже если человек, не являющийся получателем, откроет письмо, то он не увидит текст, написанный невидимыми чернилами. А если этот человек получит зашифрованное сообщение, написанное обычными чернилами, то он сможет попытаться расшифровать его. В этом главное отличие стеганографии от криптографии. Время шло, и компьютерная техника развивалась. Теперь стеганографию научились использовать в компьютерном мире. Но стеганография используется не только для передачи данных. Часто её используют для защиты от нелегального копирования. Например, редакторы сайта Genius.com, использовали два вида апострофов в текстах, чтобы вместе они составляли словосочетание red handed (с поличным) на азбуке Морзе. Водяные знаки тоже относятся к стеганографии. Но методы стеганографии также применяют и злоумышленники. В файле можно спрятать не только текст, а часть программы, при этом файл не будет считаться вирусным. Несколько лет назад злоумышленники распространяли через рекламные сети необычный баннер. Он выглядел, как цветной прямоугольник. Но самое главное, что этот баннер содержал скрипт, который распознавал цвет пикселей на картинке и записывал его в виде букв и цифр. Код, который получился из картинки, перенаправлял пользователя на сайт злоумышленников. Там жертву ждал троян, притворившийся обновлением Adobe Flash Player [1].

Моя программа создана для скрытия или извлечения текста из изображения и сравнения файлов для того, чтобы узнать, содержит изображение скрытый текст или нет. При помощи этой программы люди могут проверять на безопасность подозрительные файлы или проверять подлинность файла. Большинство других продуктов используют только один метод — LSB. Моя программа будет предлагать пользователю выбор из двух методов.

**Анализ технических требований и уточнение спецификаций**

Стеганография подразумевает под собой работу с файлами, в данном случае с изображениями, и работу с битами. Файлы изображений могут содержать большое количество байт, поэтому программа должна быстро обрабатывать большие объёмы данных.

Исходя из этих требований я выбрал язык с++, потому что с++ быстрый и низкоуровневый язык программирования, который позволяет работать с памятью и потоками данных. Ещё в нём есть типы данных, которые идеально подходят для работы с битами информации. При работе с файлами с++ позволяет перемещать курсор в любое место, что сильно облегчает считывание информации из файла.

Технология программирования — объектно ориентированное программирование, потому что этот подход обеспечивает максимальную эффективность разработки, лёгкую поддержку и изменение отдельных компонент программы, избавляет от дублирования кода. Код, написанный в парадигме объектов легче читать.

Среда разработки — Visual Studio, потому что это удобный и многофункциональный инструмент, который использует большинство разработчиков. Visual Studio имеет множество предустановленных функций и компилятор, которые нужны каждому разработчику.

**Метод встраивания в заголовок**

Этот метод самый простой в реализации, относящийся к компьютерной стеганографии: используется особенность формата данных в компьютере. В чём заключается его идея? В области памяти компьютера всегда есть зарезервированные биты. Это могут быть, например, биты между заголовком и наполнением файла, которые всегда пустые (заполнены 0-ми). В заголовках файлов есть биты, которые очень редко используются, или такие биты, при изменении которых пользователь не заметит разницы. Пример строения bmp файла. Первые 14 байт – заголовок файла (выделен синим на рисунке 1). В него входят тип файла, его размер и зарезервированные байты.

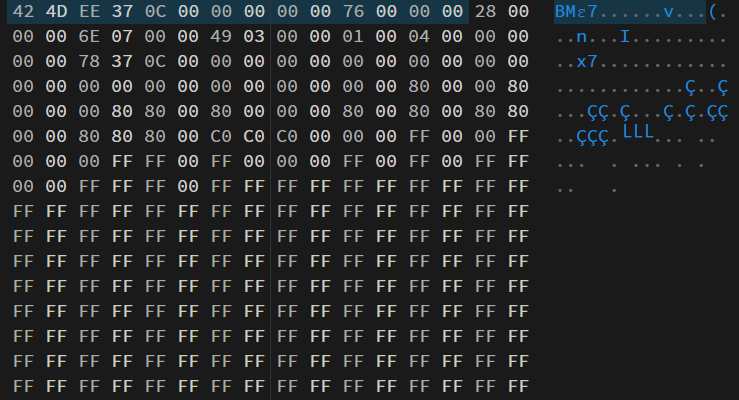


Рисунок 1. Заголовок .bmp.

Потом идёт заголовок растровой информации – 40 байт (в основном). В него входят такие данные, как размер изображения (высота и ширина), количество пикселей, тип сжатия, разрешение (горизонтальное и вертикальное), число цветовых индексов, размер информационного заголовка. Эта область выделена синим на рисунке 2.

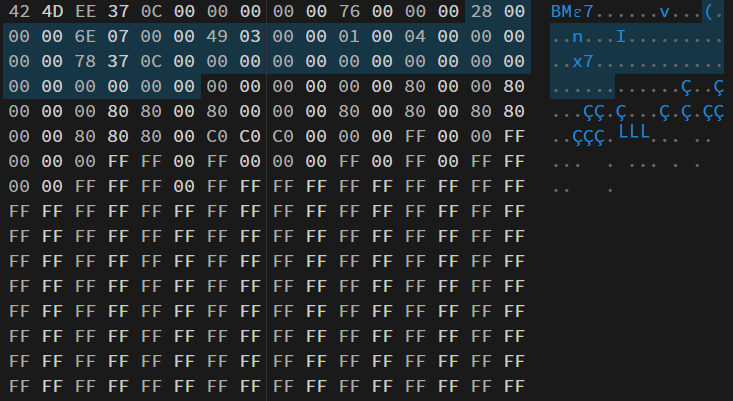


Рисунок 2. Заголовок растровой информации.

Дальше идёт цветовая палитра и растровые данные, но они не нужны.

Можно записывать текст в биты с шестого по девятый, но тогда бы длина входного текста была бы очень маленькой. Поэтому лучше записывать текст в биты с 22 по 40. Если изменить эти биты, то изображение всё также будет открываться и нормально работать, и человек не заметит никаких изменений в картинке. Но у этого метода есть минус – его очень просто обнаружить. Если после записи слова в изображение (в данном случае - «text», как на рисунке 3), открыть его любым текстовым редактором, то его можно спокойно увидеть.

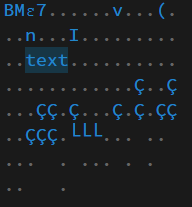


Рисунок 3. Результат встраивания в заголовок.

Но далеко не каждый человек будет каждый раз открывать изображение в текстовом редакторе, чтобы проверить, нет ли в нём скрытой информации. При использовании метода «встраивания в заголовок» ставка делается как раз на это. Этот метод очень прост в реализации. Написание программы не занимает много времени. Всё что надо знать программисту – это то, как прочитать и записать информацию в файл. Если человек хочет попробовать себя в стеганографии, то ему стоит начать именно с этого способа.

Алгоритм работы программы.

Сначала надо открыть файлы: изображение-оригинал, текстовый файл. Создать файл-результат. Потом надо отложить байты, относящиеся к заголовку. Читать байт с оригинала и записывать его в результат без изменений. Как только программа дошла до 22 байта, то первый цикл заканчивается. Второй цикл состоит в том, что он записывает в файл-результат не оригинальный байт, а заменённый на байт из файла с текстом. Когда текст закончился, начинается третий цикл. Он заполняет файл-результат оставшимися байтами из оригинала. Сложность алгоритма – линейная.

**Метод сравнивания файлов**

Этот метод не относится к стеганографии, но всё равно является очень нужным. Если есть файл-оригинал и файл, который надо проверить на скрытый текст, то можно просто сравнить их. Открываются два файла на чтение. Потом циклом берётся байт из каждого файла. Эти байты сравниваются друг с другом. Если какие-то байты отличаются, то программа выведет эти байты. Это поможет узнать, чем отличается второй файл от первого. Если над файлом проводились какие-либо операции, то некоторые байты будут отличаться в любом случае, а это значит, что проверяемый файл не является копией оригинала.

В современном мире используется другой способ проверки файла на то, является ли он копией оригинала. Если компания хочет разместить своё, например, приложение, то она выкладывает ещё и хеш-сумму. И в основном сравниваются эти хеш-суммы. хеш-сумма – результат работы хеш-функции. Хеш-функции преобразуют входные данные в код определённой длины притом если изменить во входных данных хоть один символ, то результат функции изменится кардинально. Если хеш-сумма скачанного с неофициального сайта приложения не равна хеш-сумме оригинала, то в скачанном приложении есть какие-то изменения.

В данном случае лучше не использовать хеш-функции, потому что из хеш-суммы нельзя получить файл обратно, а человеку надо узнать, где может быть стеганограмма. Хеш-функции работают, как мясорубки, то есть работают только в одну сторону. Сравнив хеш-суммы, можно узнать, одинаковы ли файлы, а если они разные, то нельзя найти отличия, поэтому лучше проходить по файлам циклом.

**Метод LSB (least significant bit)**

Этот метод гораздо надёжнее, чем встраивание в заголовок, но его реализация требует больше усилий. Он относится к цифровой стеганографии: изображение имеет намного больше информации, чем воспринимают человеческие органы чувств. Сообщение теперь будет встраиваться не в заголовок файла, а в блок, где закодированы пиксели. Каждую букву сообщения можно представить в виде 8-ми бит, каждый пиксель – один байт, следовательно, пиксель тоже можно представить в виде 8-ми бит. Теперь надо вставить часть буквы в изображение. В каждый пиксель мы вставим по одному биту нашего текста. Но какой бит менять? Так как каждый цвет – это какое-то число, то его можно изменить. Если в этом числе менять старшие разряды, то оно будет сильнее отличаться от оригинала, а если младшие, то наоборот, поэтому в пикселе надо менять последний бит. Человеческий глаз может различать около 10-ти миллионов цветов, а компьютер может закодировать около 16-ти миллионов цветов, поэтому, изменив последний бит в пикселе человек не сможет заметить такой маленькой разницы. Например, на рисунке 4 представлены два разных цвета.

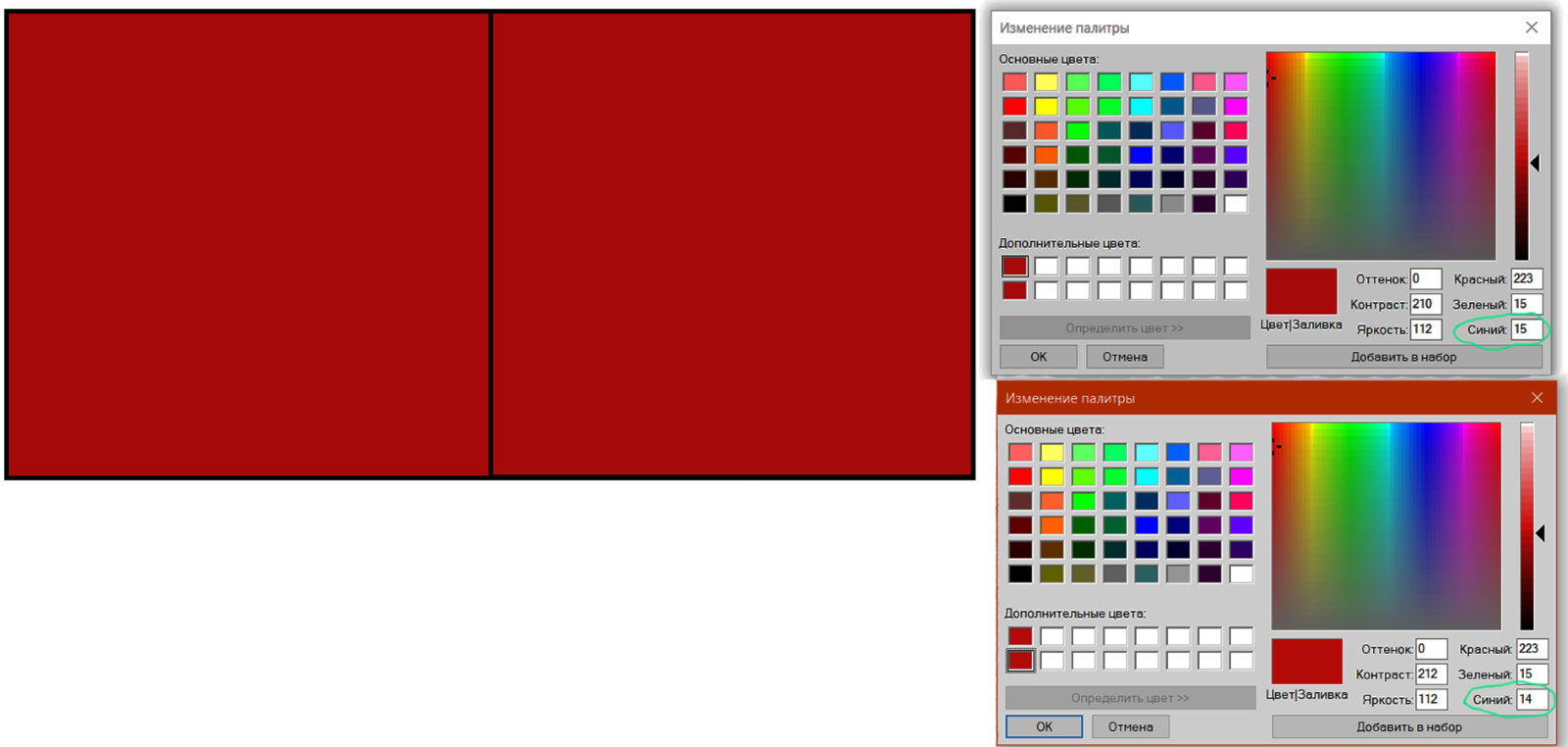


Рисунок 4. Менее значимые биты.

Этот метод невозможно обнаружить с помощью текстового редактора, в отличие от первого метода. Его можно обнаружить только перебором комбинаций последних битов, поэтому метод LSB считается самым популярным.

Для реализации этого метода надо сначала представить текст в виде бит. Потом изменять значения последних бит картинки на биты текста. Сложность алгоритма – линейная. Этот метод можно ещё усложнить, чтобы прочитать сообщение было ещё сложнее. Текст можно зашифровать. Хоть стеганография и криптография – разные науки, они часто используются вместе. Даже если человек сможет найти спрятанный текст, ему придётся его расшифровать. Ещё можно выбрать бит, начиная с которого будет происходить запись текста, например, записывать с середины изображения. Также можно придумать алгоритм записи битов, например, записывать биты через один. В общем, вариантов можно придумать бесконечно много, и все они будут усложнять получение скрытого текста.



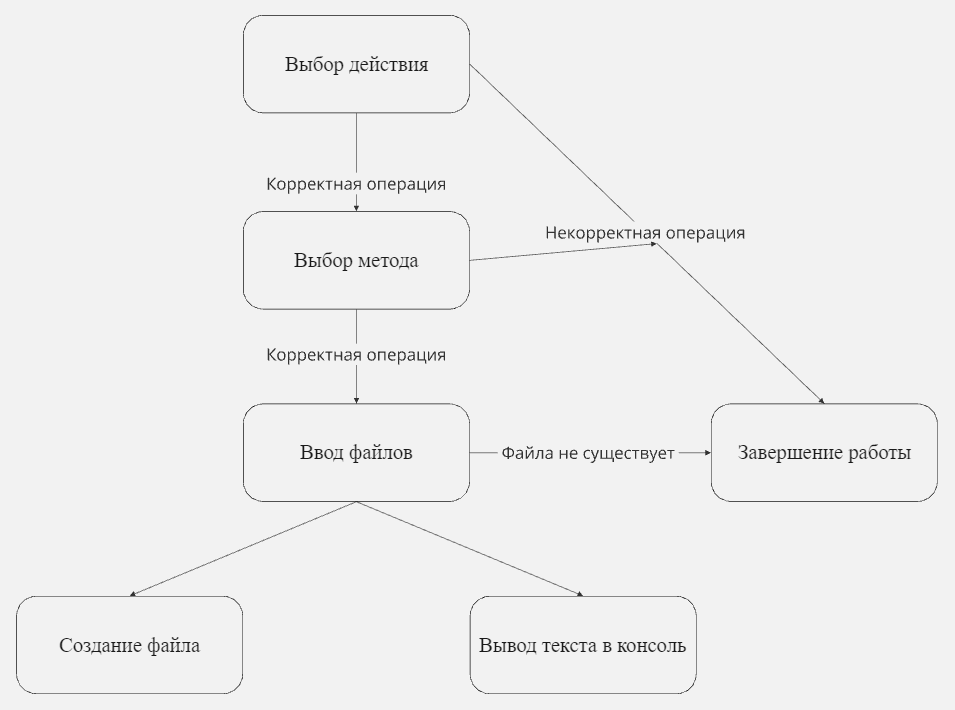
Рисунок 5. Результат LSB.

На рисунке 5 можно заметить, что текст, встроенный в картинку, невозможно распознать. Единственный способ узнать, что в изображение что-то вписано – сравнить файл с оригиналом.

**Проектирование структуры и компонентов программного продукта**

Программа имеет консольный интерфейс. Пользователь видит текстовое меню (иерархия этого меню на рисунке 6) и выбирает команду вводя число. Выбор файлов осуществляется написанием в консоль названия файла. Для такой программы консольного интерфейса будет более, чем достаточно, поэтому графического интерфейса нет. Диаграмма взаимодействия пользователя с интерфейсом представлена на рисунке 7. Также консольный интерфейс универсальный и быстрый. Результат выполнения команд может выводиться как в консоль, так и в новый файл. Это зависит от самой команды: если нужно найти в изображении текст, то этот текст выводится в консоль, если нужно спрятать текст в изображении, то создастся новый файл. Для использования программы на различных устройствах (под операционной системой Windows) создан .exe файл.

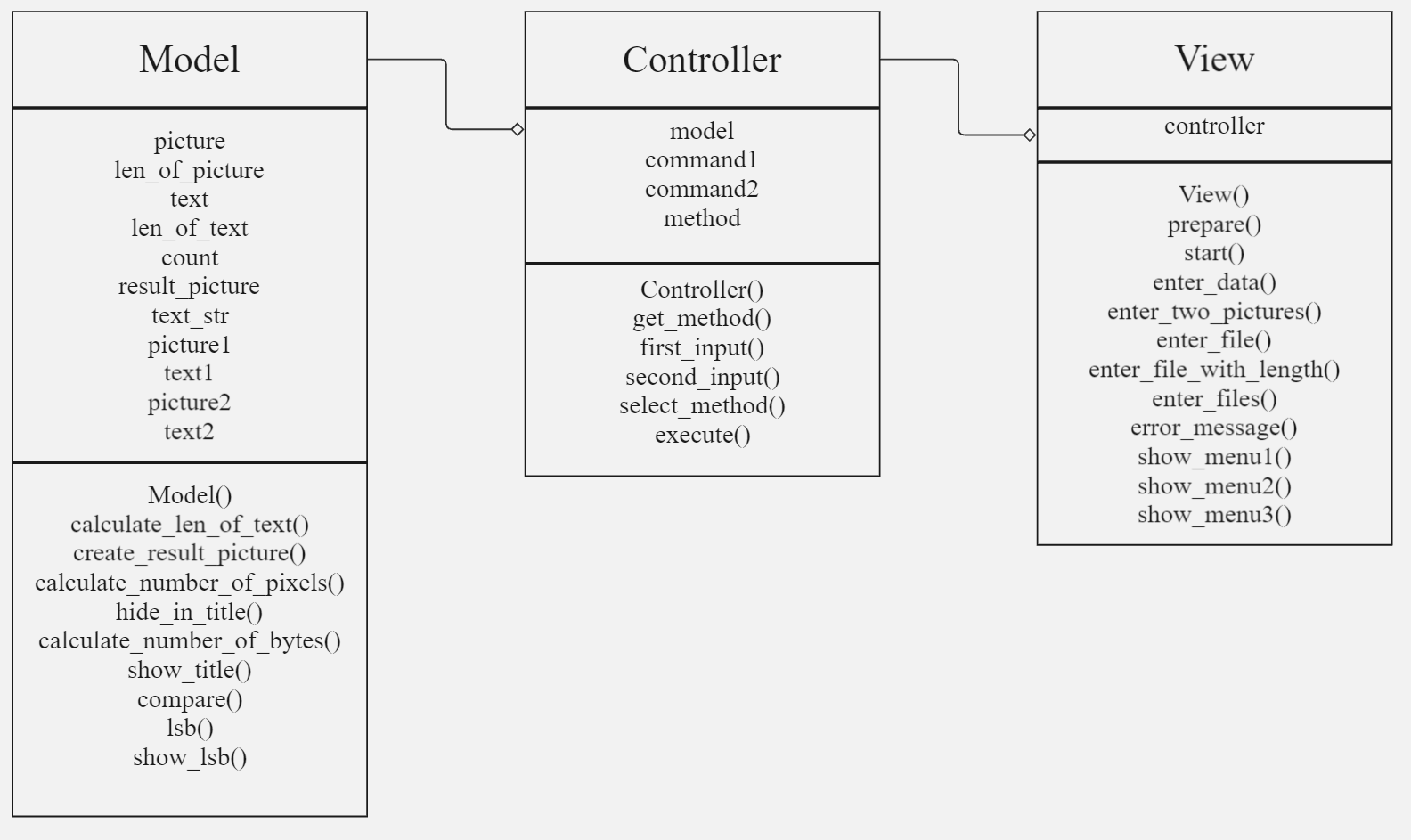
Рисунок 6. Иерархия меню.

Рисунок 7. Диаграмма взаимодействия с интерфейсом.

Программа состоит из трёх больших функциональных элементов: model, view, controller. Такой паттерн проектирования имеет название MVC. Этот шаблон разделяет архитектуру программы на три блока:

* Model (модель) – блок, содержащий основную логику программы, данные и методы для работы с ними. Модель принимает команды от контроллера и как-то изменяет данные, а потом передаёт эти данные обратно в контроллер.
* Controller (контроллер) – блок, связывающий модель и представление. Он обрабатывает действия пользователя, проверяет входные данные и передаёт их модели.
* View (представление) – блок, который отвечает за считывание данных, вывод данных, отображение интерфейса.

Модель не зависит от контроллера и представления. Контроллер содержит в себе ссылку на модель и может вызывать её методы, но не зависит от представления. Представление содержит в себе ссылку на контроллер и может вызывать его методы. Такая архитектура позволяет изменить логику одного из блоков, и при этом эти изменения не коснутся других блоков. Она представлена на рисунке 8. Также логика программы делиться на несколько блоков, которые хранятся отдельно друг от друга. Паттерн MVC позволяет упростить код большого приложения, сделать его структурированным, упростить поддержку и тестирование.

Рисунок 8. Диаграмма классов

Класс модели самый большой по объёму, потому что в нём находится больше всего полей и методов для реализации алгоритмов. Поля модели связаны с файлами, например, picture – это файл изображения, которое ввёл пользователь, len\_of\_picture – соответственно его длина. Для разных методов нужны разные файлы, поэтому в модели много полей, которые хранят разные файлы. Для того, чтобы алгоритмы, описанные выше, не получились огромными по объёму кода, части этих алгоритмов вынесены в отдельные методы. Это также позволяет избежать дублирования кода, например, при создании файла-результата или при подсчёте длины изображения.

Класс контроллера самый маленький, потому что у него меньше всего функций. Он должен перевести данные из строкового типа в числовой, если это необходимо, и проверить, не выходят ли эти данные за диапазон. В методе execute() контроллер вызывает нужный метод у модели по указателю на неё. Этот метод выбирается в select\_method() по входным данным.

Класс представления не хранит каких-либо данных, кроме указателя на контроллер. Он содержит только методы вывода меню и ввода данных. Когда ввод закончен, представление вызывает метод execute() у контроллера и передаёт ему входные данные.

**Выбор стратегии тестирования и разработка тестов**

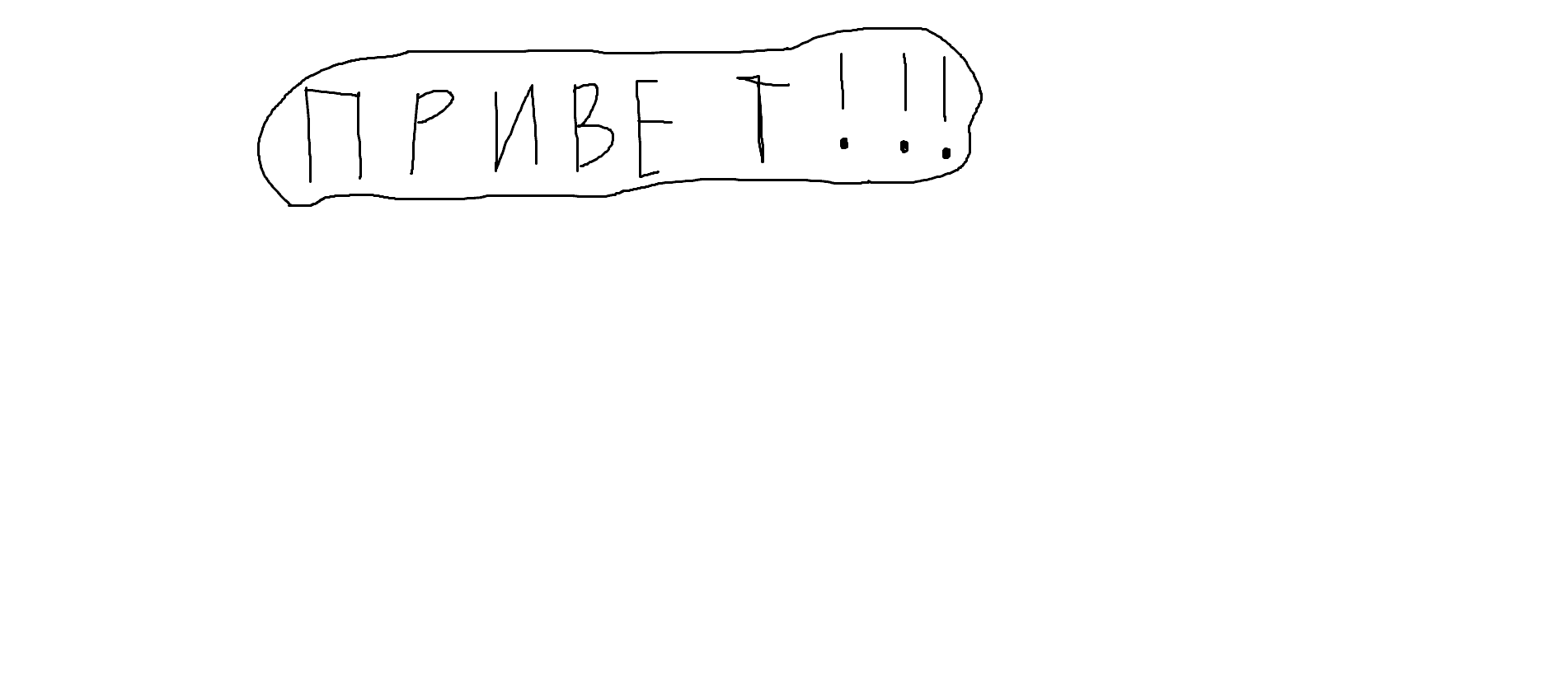
Для тестирования программы был выбран метод модульного тестирования. Метод заключается в том, что проводится тестирование одного какого-либо отдельного модуля (блока) программы. Так как основными методами в проекте являются LSB и встраивание в заголовок, то будут проводиться тесты только этих методов.

Для тестирования была выбрана библиотека gtest, реализующая unit tests. Плюсы юнит тестирования заключаются в том, что можно проверять только одну функцию или класс в программе. Это означает, что во время тестирования не придётся вводить никакие данные в консоль, достаточно передать их в виде аргументов функциям, которые будут тестироваться. После выполнения этой функции останется только сравнить результат программы и корректный ответ. Так как при написании кода я использовал паттерн MVC, то код у меня уже поделён на классы и функции, значит использование юнит тестов станет ещё легче.

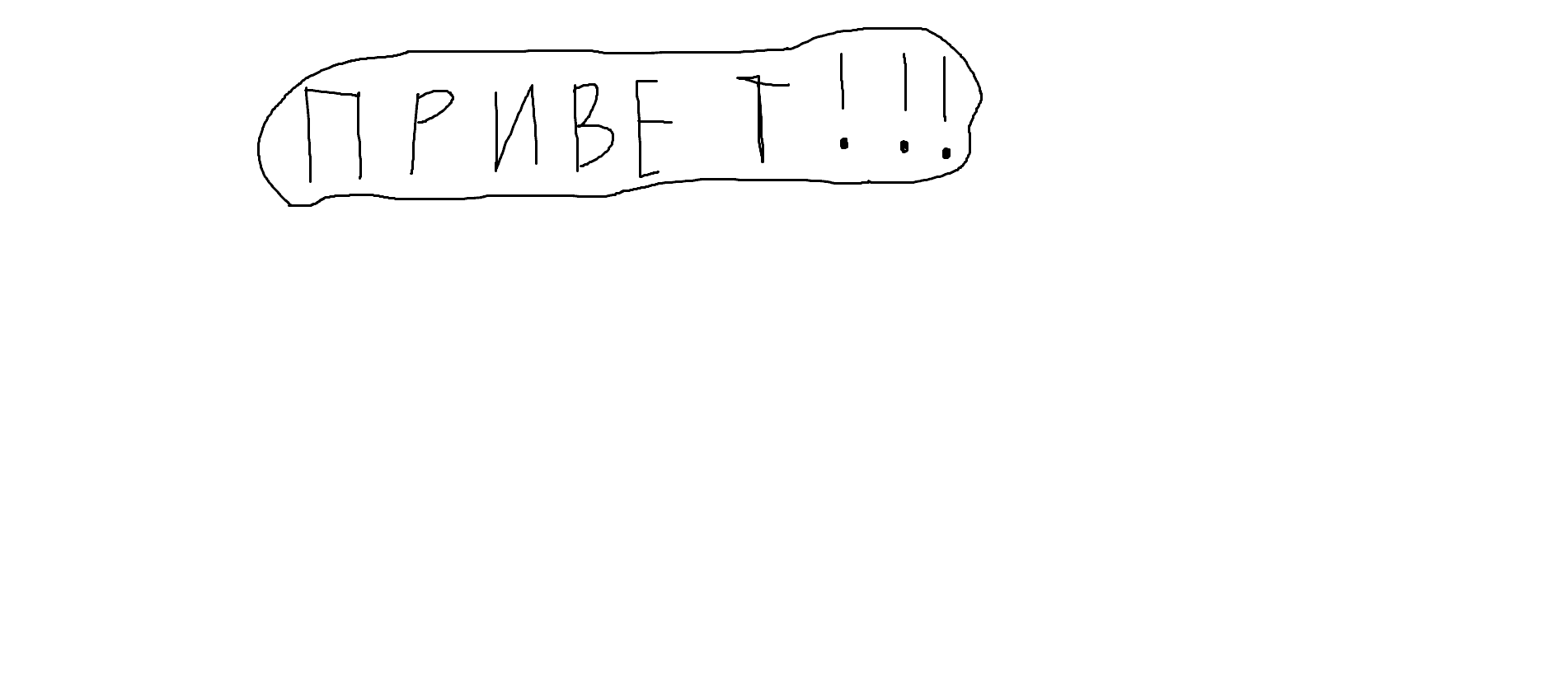
Тесты в моей программе построены по одному шаблону: есть заранее подготовленный набор изображений, в разных тестах программа использует разные изображения, потом одним из методов происходит скрытие текста, потом методом сравнивания файлов программа находит отличия в файлах, потом эти отличия сравниваются с правильными ответами.

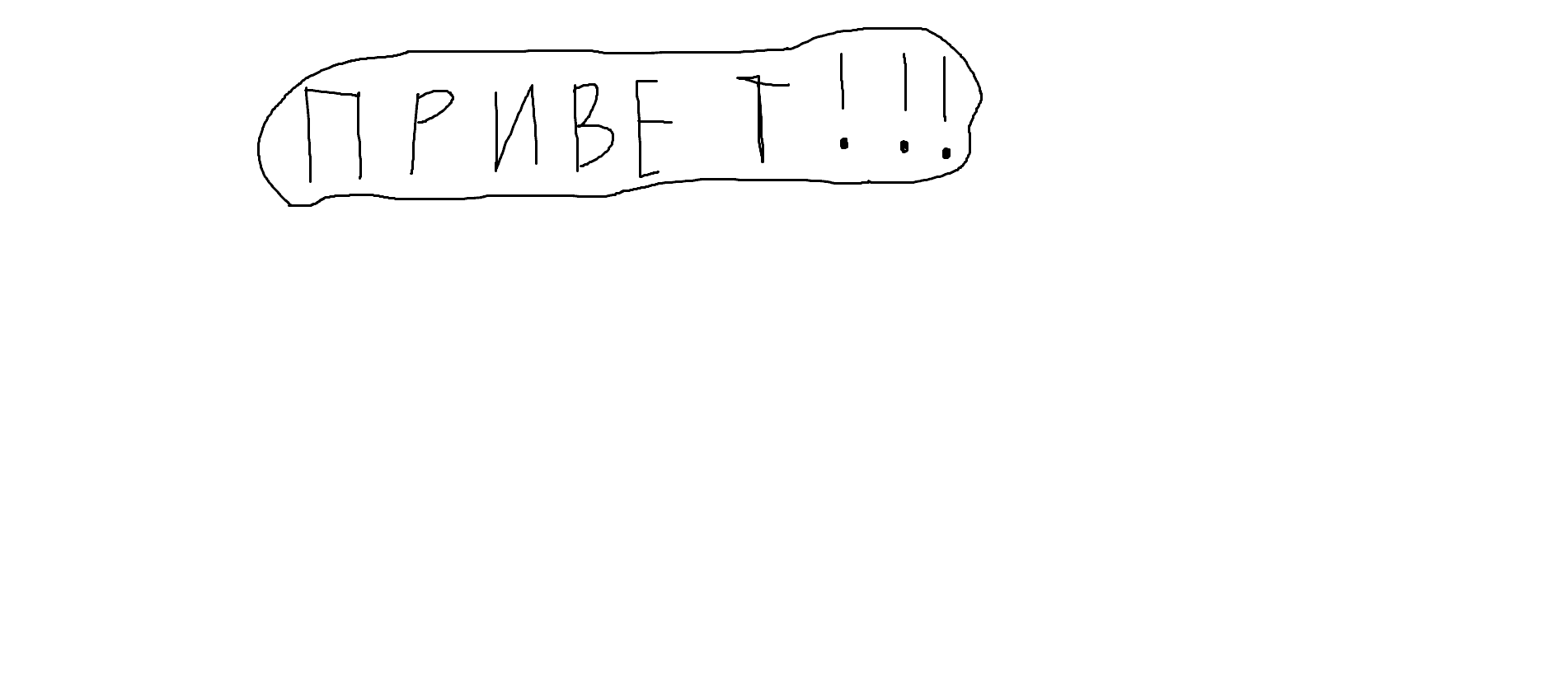
Пример.

Исходное изображение (рисунок 9):

Рисунок 9. Пример исходного изображения.

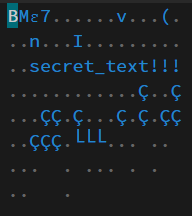
Результаты работы алгоритмов в виде изображений (рисунки 10 и 11):

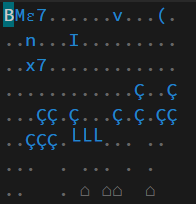
Рисунок 10. Результат встраивания в заголовок.

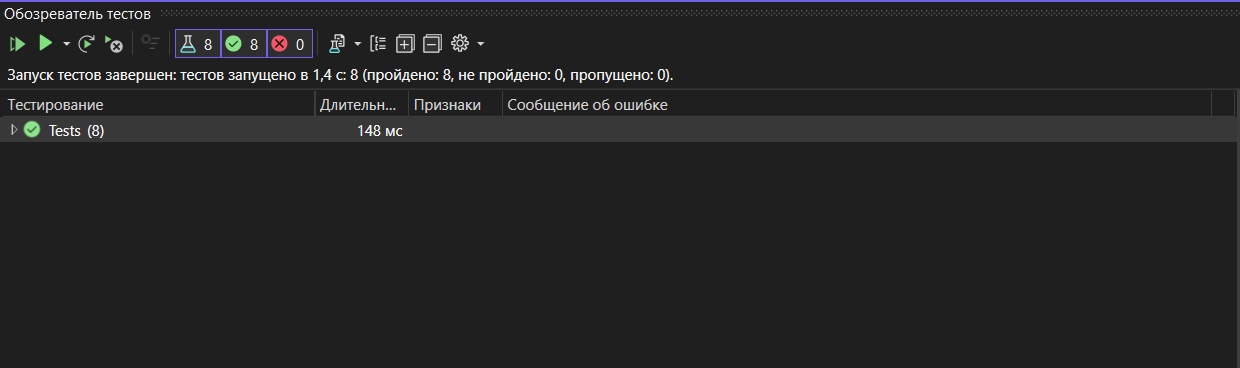
Рисунок 11. Результат LSB.

В обоих случаях встраивался текст: secret\_text!!!

Результаты работы алгоритмов в виде байтов (рисунки 12 и 13):

Рисунок 12. Байтовое представление встраивания в заголовок.

Рисунок 13. Байтовое представление LSB.

Рисунок 14. Подтверждение работы тестов.

**Заключение**

В данной работе была разработана программа реализующая алгоритмы стеганографии: встраивание в заголовок и least significant bit. В программе использовались современные подходы к программированию: объектно-ориентированное программирование, паттерны проектирования, unit tests. Язык программирования с++ обеспечил программе высокую скорость обработки файлов. Консольный интерфейс понятен и достаточно удобен, поэтому не уступает графическому интерфейсу. Алгоритмы протестированы и работают корректно. Паттерн MVC повысил читаемость кода и облегчил его изменение. Программа распространяется открытым исходным кодом и размещена на платформе GitHub, поэтому любой желающий может предложить идеи для исправления и улучшения.

Благодаря моей программе любой желающий теперь может скачать её код с GitHub и, при желании, настроить под себя, тем самым повысив свою безопасность в Интернете. Программа позволит как обнаружить что-то вредоносное в изображении, так и скрыть какую-нибудь информацию от прослушивания.

**Литература**

1. В фейке Adobe Flash Player маскируется вирус-майнер – AltCoinLog – URL: <https://altcoinlog.com/adobe-flash-player-virys-mainer/> (дата обращения: 02.10.2024) – Текст: электронный. Режим доступа: свободный.

2. Двоичная тайнопись (по материалам открытой печати) | КомпьютерПресс – URL: <https://compress.ru/article.aspx?id=10416> (дата обращения: 02.10.2024) – Текст: электронный. Режим доступа: свободный.

3. Стеганография – «Информационная безопасность» - URL: <http://security.pmkb.sfu-kras.ru/blog/steganografiya/> (дата обращения: 03.10.2024) – Текст: электронный. Режим доступа: свободный.

4. День CIH.win9x / Хабр – URL: <https://habr.com/ru/post/91938/> (дата обращения: 04.10.2024) – Текст: электронный. Режим доступа: свободный.

5. Чернобыль – самый страшный вирус – URL: <https://www.youtube.com/watch?v=jL5zSM_8WJk> (дата обращения: 04.10.2024) – Видео. Режим доступа: свободный.

6. Aperi’Solve – URL: <https://aperisolve.fr/> Режим доступа: свободный.

7. Steganography Online – URL: <http://stylesuxx.github.io/steganography/> Режим доступа: свободный.

8. Подробный формат BMP – URL: <https://russianblogs.com/article/5252602914/> (дата обращения: 10.10.2024) – Текст: электронный. Режим доступа: свободный.

9. Методы стеганографии, используемые для сокрытия информации – URL: <https://masters.donntu.org/2019/fknt/sidorchuk/library/transarticle.htm> (дата обращения: 10.10.2024) – Текст: электронный. Режим доступа: свободный.

10. Что такое стеганография? | Блог Касперского – URL: <https://www.kaspersky.ru/blog/digital-steganography/23025/> (дата обращения: 10.10.2024) – Текст: электронный. Режим доступа: свободный.

11. Ссылка на мой репозиторий – URL: <https://github.com/Timur1414/steganography> Режим доступа: свободный.