INFORME DESAFIO I

ANYELA MARTÍNEZ ORTEGA

AUGUSTO ENRIQUE SALAZAR JIMENEZ

INFORMÁTICA II

UNIVERSIDAD DE ANTIOQUIA
FACULTAD DE INGENIERÍA

ABRIL DE 2025

Análisis del problema y consideraciones para la alternativa de solución propuesta. Desde el principio entendí que el problema era más complejo de lo que parecía. No solo se trataba de aplicar operaciones a nivel de bits, sino también de identificar cuál operación específica se había aplicado a una imagen sin saberlo previamente. Eso hizo que todo fuera más retador, porque había que pensar en un algoritmo que pudiera probar automáticamente varias transformaciones y verificar si alguna coincidía.

La alternativa de solución que propuse fue organizar el proceso de una forma muy estructurada: primero implementar las funciones necesarias (XOR, rotaciones y desplazamientos de bits), después crear un sistema que aplicara esas funciones una por una y, en cada intento, comparar el resultado con el archivo de referencia (txt). Esto permitía saber en qué momento se encontraba la transformación correcta.

Una consideración importante fue que no busqué hacer todo el procedimiento completo en un solo paso gigante, sino que decidí hacerlo de forma controlada, paso a paso, para asegurar que en cada transformación los datos coincidieran exactamente. Creo que esa fue la clave para lograr una solución práctica y que realmente funcionara.

Además, aprendí que el orden, la paciencia y la verificación constante son fundamentales en este tipo de problemas donde hay muchos pequeños detalles que pueden afectar el resultado final.

Esquema donde describa las tareas que usted definió en el desarrollo de los algoritmos. Durante el desarrollo de la solución, estructuré el trabajo en las siguientes tareas:

1. Estudio de conceptos clave: Realicé una revisión detallada de conceptos fundamentales para el ejercicio, como operaciones a nivel de bits, manejo de punteros, arreglos dinámicos en C++, y comprensión de imágenes en formato BMP. Este conocimiento fue esencial para construir una solución adecuada.
2. Análisis de los archivos base proporcionados: Comprendí a profundidad el contenido y propósito de los recursos entregados por los docentes, incluyendo el código inicial, las máscaras, la imagen enmascarada y los archivos de texto que contenían las semillas de enmascaramiento.

3. Desarrollo de funciones auxiliares: Implementé primero las funciones necesarias para las operaciones de transformación (XOR, rotaciones, desplazamientos de bits, aplicación de máscara y comparación de resultados). Disponer de estas funciones facilitó la organización posterior del algoritmo principal.
4. Estructuración del flujo del programa: Organicé el código de manera que cada función se aplicara en el orden correcto, permitiendo así identificar y revertir las transformaciones aplicadas a la imagen original.
5. Validación mediante casos de prueba: Utilicé los casos de prueba proporcionados para verificar el funcionamiento del programa. Realicé ajustes iterativos hasta lograr que el código resolviera correctamente cada caso.
6. Control de versiones: Realicé commits frecuentes en el repositorio, asegurándome de registrar cambios relevantes y mantener un historial claro del progreso alcanzado.
7. Preparación de la entrega: Finalmente, consolidé todo el material trabajado, asegurando el cumplimiento de los requisitos del desafío y la correcta documentación de la solución.

Algoritmos implementados.

Para desarrollar el programa que resuelve el desafío de reconstruir la imagen original, implementé varios algoritmos específicos que trabajan de manera conjunta. A continuación, explico cada uno de ellos:

- Aplicación de operaciones de transformación

Teníamos que identificar qué operación de transformación había sido aplicada a la imagen original. Para esto, creé los siguientes algoritmos:

Operación XOR (OperatorXOR): Realiza una operación bit a bit entre la imagen de entrada y una imagen máscara. Cada componente R, G y B de los píxeles es procesado aplicando el operador ^ (XOR).

Rotación de bits (Rbits): Implementé un algoritmo que rota los bits de cada componente RGB a la derecha. Probé todas las rotaciones posibles de 1 a 7 posiciones para encontrar una posible coincidencia.

Desplazamiento de bits (Dbits): Desarrollé una función que desplaza los bits hacia la derecha o hacia la izquierda (dependiendo de la dirección solicitada), también probando desplazamientos de 1 a 7 posiciones.

Cada una de estas funciones toma la imagen original y genera una nueva imagen transformada que puede ser comparada con el archivo de pista (el archivo .txt).

- Aplicación de la máscara (ApliMask)

Luego de aplicar cada posible transformación, diseñé una función que simula el enmascaramiento. Esta función suma, píxel por píxel, la imagen transformada con la máscara entregada, a partir de una posición semilla (offset). El resultado es un nuevo conjunto de datos RGB que representa el archivo de pista que debía obtenerse.

- Comparación de resultados (compareTXT)

Implementé un algoritmo de comparación que verifica si el resultado obtenido después de aplicar la transformación y la máscara coincide exactamente con el archivo de pista entregado. Esto me permite identificar cuál operación (y en qué cantidad) fue la que se aplicó originalmente.

- Control de memoria dinámica

Al tratar directamente con arreglos dinámicos (sin usar estructuras ni STL), fue fundamental implementar buenas prácticas de liberación de memoria. Cada vez que creaba un nuevo arreglo o terminaba de usarlo, me aseguraba de liberar correctamente el espacio utilizando delete[], evitando fugas de memoria y garantizando la eficiencia del programa.

Problemas de desarrollo que afrontó.

Durante el desarrollo del ejercicio enfrenté varios problemas que, en más de una ocasión, me hicieron pensar en desistir de encontrar una solución. Uno de los mayores retos fue

estructurar correctamente el código para que pudiera identificar qué operación se había aplicado a la imagen enmascarada. Aunque tenía muchas ideas, ninguna funcionaba como esperaba. Finalmente, después de varios intentos y ajustes, logré implementarlo correctamente, aunque fue un proceso bastante complejo y exigente.

Otro problema importante surgió por errores en el nombramiento de los archivos. Debido a pequeños fallos en los nombres, los casos de prueba no funcionaban correctamente: los archivos .txt nunca coincidían, lo que me llevó a pensar en algún momento que todas las funciones que había desarrollado estaban mal. Esto generó bastante frustración, pero con paciencia fui depurando y entendiendo que el problema era externo a la lógica de mi código.

Además, durante la mitad del desafío, el programa de Qt dejó de funcionar correctamente. El programa no leía el código ni ejecutaba nada, lo que me obligó a invertir bastante tiempo en resolver este inconveniente técnico. Afortunadamente, logré solucionarlo y continuar con el proyecto.

En conclusión, tuve que superar varios obstáculos durante el proceso y con esto pude corregir los errores y poco a poco lograr una solución sólida para el desafío.

Evolución de la solución y consideraciones para tener en cuenta en la implementación.

Finalmente, logré encontrar una solución funcional al desafío. Sin embargo, es importante aclarar que el algoritmo no realiza todo el proceso de transformación en un solo intento automático. Es decir, si a la imagen original se le aplicaron varias operaciones (por ejemplo, siete transformaciones distintas), entonces manualmente se deben preparar los recursos (imágenes, máscaras y archivos .txt) y ejecutar el algoritmo varias veces, una por cada paso.

De esta manera, en cada ejecución se identifica correctamente cuál fue la operación aplicada en ese paso específico. Considero que esta es una forma más sencilla y clara de implementar la solución, ya que permite un mejor control sobre el proceso y asegura que en cada etapa los archivos de comparación (los .txt) coincidan adecuadamente, minimizando los errores y facilitando la verificación de los resultados.

Esta estrategia también ofrece la ventaja de detectar de manera precisa qué operación corresponde a cada transformación, evitando confusiones o pasos mal aplicados.