INFORME FINAL

Desafío 1 - Procesamiento de Imágenes BMP

Informática II - 2025-1

Autor: Franco José Carrascal Montes

Introducción

El presente informe describe el proceso de desarrollo de un programa en lenguaje C++ (usando Qt) para reconstruir imágenes BMP de 24 bits que han sido sometidas a transformaciones a nivel de bits, como rotaciones, desplazamientos y operaciones XOR, siguiendo las condiciones planteadas en el Desafío 1 de la materia Informática II.

1. Análisis del problema

A. El problema planteado consiste en recuperar una imagen original a partir de su versión transformada, sin conocer el orden ni el tipo de transformaciones aplicadas. Para ello, se proporciona:

Una imagen modificada o transformada (I_D.bmp).

Una imagen de distorsión (I_M.bmp) usada para operaciones XOR.

Una máscara (M.bmp) utilizada para enmascarar partes de la imagen.

Archivos .txt que contienen resultados de enmascaramiento en distintos pasos.

Se requiere revertir cada transformación, validando en cada etapa mediante las máscaras.

El reto implica realizar un proceso de ingeniería inversa sobre las imágenes usando operaciones de bajo nivel (bit a bit). Considero que la manera en que lo entendí yo es un poco más fácil y no tan arriesgada debido a mi poco conocimiento, además de que me ayudé mucho con videos y otras herramientas. Además pienso que puede haber una y mil formas mas de realizarlo y también de entender el desafío, porque es algo que nos paso a todos los compañeros, muchos tomamos esto de una manera un poco diferente.

B.

- Cargar imágenes BMP en arreglos dinámicos.
- Implementar operaciones bit a bit: XOR, rotaciones y desplazamientos.
- Leer archivos de enmascaramiento y verificar la integridad de los datos.
- Detectar y revertir transformaciones paso a paso.
- Reconstruir la imagen original asegurando la correcta gestión de memoria.

C. **Algoritmos implementados**

Operaciones bit a bit:

XOR entre imágenes.

Rotación de bits hacia la derecha.

Rotación de bits hacia la izquierda.

Desplazamiento de bits hacia la derecha.

Desplazamiento de bits hacia la izquierda.

Todas estas funciones trabajan directamente con arreglos dinámicos de tipo unsigned char*.

Carga de imágenes BMP:

Se utilizó QImage para importar imágenes en formato RGB888, copiando los datos de píxeles sin estructuras ni STL.

Carga de archivos de enmascaramiento:

Se implementó una función para leer el desplazamiento (semilla) y los valores RGB del enmascaramiento desde archivos .txt, almacenándolos en arreglos dinámicos unsigned int*.

Verificación de máscara:

Se desarrolló una función que compara si la suma de los valores de la imagen transformada más la máscara coincide con los valores del archivo .txt.

Reconstrucción de la imagen:

El algoritmo invierte las operaciones en orden inverso, en cada paso:

Desenmascara usando la máscara y los valores del .txt.

Detecta y revierte la transformación realizada.

Valida la corrección del desenmascarado.

Exporta imágenes intermedias para depuración.

Problemas de desarrollo encontrados

Manejo manual de memoria dinámica (new y delete[]) sin estructuras ni STL. A pesar de no saber usaralas de cierta forma se ve que es un poco mas sencillo, pero aun así es difícil para alguien que tiene poco tiempo en este lenguaje

Validación correcta del desenmascaramiento para asegurar que las operaciones inversas fueran exitosas.

Organización del código para evitar duplicación de funciones y facilitar la depuración.

No había explicación por donde tomar el desafío, que hacer primero etc.

E**volución de la solución**

- Inicio del proyecto cargando y exportando imágenes BMP.(Con el código ya implementado que se nos proporciono)
- Implementación de operaciones básicas bit a bit (XOR, rotaciones, desplazamientos).
- Lectura y validación de archivos de enmascaramiento.
- Creación de funciones de desenmascarado y verificación.
- Desarrollo del algoritmo completo de reconstrucción y validación.
- Depuración y generación de imágenes intermedias para facilitar el análisis.

Conclusiones

El desafío permitió consolidar habilidades fundamentales en C++ tales como el manejo de punteros, memoria dinámica, operaciones bit a bit, y la organización modular de código. Aunque es cierto que este desafío fue un poco confuso debido a que en el tiempo que se implementó aun teníamos poco conocimiento a detalle del desafío es difícil aun así documentándose y utilizando las múltiples herramientas que nos podrían ayudar

Además, reforzó la importancia de un proceso iterativo de prueba y validación en problemas de ingeniería inversa.