**Informe inicial desafio I**

**Hawer Hernandez Vargas**

**Andres Felipe Henao Velasquez**

**2025**

**Introducción**

Este desafío simula un escenario de ingeniería inversa sobre imágenes BMP transformadas, en el cual el estudiante debe reconstruir una imagen original que ha sido alterada mediante varias transformaciones a nivel de bits (como XOR, desplazamientos, rotaciones) y técnicas de enmascaramiento con una máscara RGB.

La dificultad reside en que el orden y tipo de transformaciones aplicadas no se conoce. Solo se tiene como pista una serie de archivos .txt que registran el resultado del enmascaramiento en cada etapa.

El ejercicio pone a prueba la lógica, la capacidad de análisis, y especialmente el dominio de punteros, arreglos dinámicos, y manipulación binaria en C++ sin el uso de STL ni estructuras.

**Resumen del desafío**

**Objetivo General**

Reconstruir una imagen original BMP de 24 bits que fue transformada y enmascarada en varias etapas, usando solo los resultados finales y los archivos de rastreo disponibles.

**Recursos proporcionados**

Código fuente base para:

* Leer imágenes BMP.
* Leer archivos .txt de enmascaramiento.
* Exportar imágenes desde arreglos dinámicos.
* Dos casos de prueba con imágenes de entrada y salida.

**Transformaciones posibles**

Operaciones a nivel de bit:

* XOR
* Desplazamiento (hasta 8 bits)
* Rotación (hasta 8 bits)
* Enmascaramiento con una máscara M (selección de porciones aleatorias y suma con la máscara).

**Actividades para realizar**

* [15%] Implementar funciones para XOR, desplazamientos y rotaciones.
* [10%] Realizar pruebas y análisis sobre estas transformaciones.
* [10%] Validar resultados del enmascaramiento con los .txt.
* [65%] Deducir el orden y tipo de transformaciones para reconstruir la imagen original.

**Requisitos obligatorios**

* Implementar en C++ con Qt.
* Usar memoria dinámica, arreglos y punteros.
* No usar STL ni struct.
* Crear un repositorio público con commits frecuentes.
* Generar un informe técnico con análisis, algoritmos y evolución.
* Grabar un video (5 a 11 minutos) explicando:
* La solución.
* Demostración.
* Código y decisiones técnicas.
* Subir ambos enlaces (repositorio y video) a Ude@.
* Asistir a sustentación obligatoria.

**Fechas clave**

* 12 de abril: entrega del análisis y diseño.
* 25 de abril: entrega final de implementación.

**Análisis del problema:**

Consiste en transformar una imagen por medio de operaciones bit a bit las cuales son: XOR, desplazamiento y rotación en un orden desconocido para después analizar las transformaciones, verificar resultados para identificar las operaciones aplicadas con el fin de reconstruir la imagen original en base a las operaciones de transformación de la imagen lo cual nos va a permitir revertir sus transformaciones.

**Diseño de solución:**

En principio se ha desarrollado una idea para dar solución al problema el cual hemos estructurado en 5 fases las cuales hemos definido de la siguiente manera:

* Lectura de datos:

El cual consiste en el código proporcionado por el profesor el cual permite cargar una imagen BMP y extrae los datos de pixeles en formato RGB, además, de crear una imagen utilizando los datos contenidos en el arreglo dinámico y también nos permite abrir los archivos .txt donde se almacena una semilla y los valores RGB resultantes del proceso de enmascaramiento.

* Transformación de la imagen:

En esta etapa vamos a crear funciones las cuales nos permite hacer transformaciones sobre la imagen original, por medio de operaciones bit a bit en un orden desconocido la idea es crear funciones para cada tipo de operación la cual es de implementar en otra función que me permita hacer las operaciones en un orden desconocido después hacer el enmascaramiento para generar los archivos .txt

* Experimentar y analizar con las diferentes operaciones:

En esta fase vamos a aplicar diferentes operaciones bit a bit sobre una imagen es decir transformar la imagen y vamos a observar cómo cambia la información, cada píxel por medio de las operaciones bit a bit.

* verificar el resultado del enmascaramiento:

Vamos a verificar si el enmascaramiento se aplico correctamente, debemos comparar la imagen transformada final, compara la máscara, los archivos .txt

Idea de código para la forma de verificación:

* + Código para cargar archivos (imagen transformada, archivos .txt)
  + Verificar píxel por píxel
* Reconstruir imagen original:

Debemos de implementar un con código que nos permita determinar que operaciones bit a bit se desarrollaron y en qué orden esto con el fin de aplicar operaciones que nos permita revertir las transformaciones para llegar a la imagen original.

Debemos de definir las operaciones bit a bit que se realizaron, como no se en que orden debemos de probar todas las operaciones usando el enmascaramiento, de forma que se implemente un código que pruebe las distintas combinaciones y orden, aplicando de forma inversa a los datos transformados usando la semilla como guía para determinar el orden.