**Informe Desafío I**

**Estudiantes**: Alejandro Bedoya Zuluaga y Esteban Rodríguez Monsalve  
**Profesor**: Aníbal José Guerra Soler

**Curso**: Informática II

Para la solución de este desafío, se plantearon varias alternativas, las cuales fueron posteriormente implementadas en el entorno de trabajo **Qt**. Este framework nos permitió, mediante el uso responsable de sus librerías, manipular imágenes de manera eficiente, además de exportarlas y guardarlas en archivos.

En la solución propuesta no se hizo uso de estructuras de datos complejas como listas, pilas o colas. En su lugar, se utilizaron punteros dinámicos, lo cual permitió gestionar la memoria de forma flexible y directa. Gracias a los punteros, fue posible almacenar y manipular la información de las imágenes de manera eficiente, accediendo a sus componentes individuales (RGB) en memoria sin necesidad de copias adicionales ni estructuras intermedias.

El uso de memoria dinámica facilitó el tratamiento de grandes volúmenes de datos, como las matrices de píxeles, permitiendo recorrer y modificar la información de forma ordenada y precisa. Además, trabajar directamente con punteros nos permitió realizar operaciones de bajo nivel, como el acceso y modificación bit a bit de los datos, fundamentales para las tareas de enmascaramiento y desenmascaramiento.

Esta estrategia, basada en el manejo manual de memoria, no solo optimizó el rendimiento del programa, sino que también reforzó nuestra comprensión de cómo se organiza y procesa la información a nivel interno en un sistema informático. Gracias a ello, logramos una realización satisfactoria de este proyecto, cumpliendo con los objetivos de manipulación, análisis y restauración de imágenes encriptadas.

La información de las imágenes fue almacenada en un arreglo de caracteres (char), aprovechando que cada carácter ocupa 8 bits, lo que facilita el tratamiento a nivel de bits. Gracias a las operaciones explicadas en clase, desarrollamos funciones específicas para trabajar directamente sobre los bits de cada componente de color. Estas funciones incluyen operaciones como **XOR**, **rotaciones** y **sumas de píxeles**, las cuales permiten desenmascarar las imágenes, liberándolas de las máscaras aplicadas.

Posteriormente, el flujo del programa continúa con la verificación de las funciones utilizadas y su orden de aplicación. Para ello, se implementaron funciones que automatizan el proceso de verificación: comparan una cierta cantidad de bits contra datos almacenados en archivos de referencia. Si la operación aplicada sobre la imagen coincide con los datos de referencia, se confirma que esa es la función utilizada en esa etapa de encriptación.

Una vez determinadas las operaciones y su orden correcto, se procede al proceso de **desenmascaramiento**. Según la operación identificada, el programa invoca su función respectiva para actuar sobre la imagen distorsionada. Este proceso se repite para cada etapa descubierta, siguiendo el orden correcto de las funciones.

Finalmente, al completar todas las etapas, la imagen queda completamente desenmascarada y el programa concluye su ejecución exitosamente.