DESAFÍO 1.

INTEGRANTES**:**

FABIANA DIAZ ANAYA

DANIELA HAYBIB CARVAJALINO MUÑOZ

DOCENTES:

DOCTOR ANÍBAL GUERRA

DOCTOR AUGUSTO SALAZAR.

INFORME SOBRE EL PROCESO DE ANÁLISIS Y DISEÑO DE LA SOLUCIÓN DEL DESAFIO 1.

UNIVERSIDAD DE ANTIOQUÍA.

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

12 DE ABRIL DE 2025

MEDELLÍN

**INTRODUCCIÓN.**

En este informe se describirá el análisis, desarrollo y la evolución de la solución a recuperar la imagen original BMP modificada mediante transformaciones bit a bit, con y sin operaciones de enmascaramiento, como parte del Desafío N°1 del curso de Informática II. La presente solución combinó la utilización de memoria dinámica, punteros, operaciones a nivel de bits y la comprobación de los mismos contra datos de archivos.txt.

**CONSIDERACIONES PARA LA ALTERNATIVA DE SOLUCION PROPUESTA**

Desde el inicio se planteó que la solución debe:

* Aplicar a los bytes de la imagen operaciones XOR rotaciones y desplazamientos.
* Implementar el uso de una máscara para validar las transformaciones aplicadas.
* Verificar los resultados mediante la comprobación con los datos de los archivos .txt.
* Permitir un acceso eficiente a la memoria dinámica.
* Permitir descartar una transformación de manera inmediata si al momento de la verificación no había coincidencia.

Inicialmente se afirmó que se debían realizar funciones por separado para cada tipo de transformación y verificación, siempre se mantuvo y respeto esta estructura, pero a medida que se resolvía se optimizo integrándose lógica.

**ESQUEMA DE TAREAS DEFINIDAS EN EL DESARROLLO DE LOS ALGORITMOS**

* Carga de Imágenes: cargar la imagen modificada, la imagen de máscara y la imagen del XOR a memoria dinámica.
* Carga de Archivos.txt: leer la semilla, las tripletas RGB de los archivos.txt.
* Procesamiento de Transformaciones: intentar aplicar las transformaciones XOR, rotaciones (izquierda y derecha) de 1 a 7 bits, desplazamientos (izquierda y derecha) de 1 a 7 bits.
* Verificación de Transformaciones: verificar que la imagen transformada y enmascarada sea la misma que los datos de los.txt
* Actualización de la Imagen: una vez que esta transformación es “correcta”, aplicarla a toda la imagen lograda hasta el momento
* Exportación de Imagen: guardar la imagen resultante como I\_D.bmp.
* Liberación de Memoria: borrar manualmente los arreglos dinámicos usados.

**ALGORITMOS IMPLEMENTADOS**

* **Loadpixel:** Carga una imagen BMP como un arreglo RGB888
* **ExportImage:** Guarda en un archivo bmp los datos de píxeles que hay en un arreglo dinámico.
* **LoadSeedMasking:** Lee un archivo txt que contiene una semilla y tripletas RGB
* **Compararvalores:**
* **VerificarXor, AplicarXor**
* **VerificarRotacion, AplicarRotacion**
* **VerificarDesplazamiento, AplicarDesplazamiento**

**PROBLEMAS DE DESARROLLO QUE SE PRESENTARON**

* **Semilla y desplazamiento:** Hubo errores al interpretar desde qué índice de la imagen se empezaba a usar el enmascaramiento, inicialmente porque tenía que depender de la semilla.
* **Verificacion incorrecta:** Al momento de aplicar la verificacion hubo un problema muy persistente ya que al momento de verificar los datos de la imagen transformada con la mascara, se estaba accediendo a lugares invalidos de la memoria.
* **Desplazamiento:** El desplazamiento no es leido por el codigo por lo que no imprime los desplazamientos realizados, pensamos que es porque hacer la verificacion a la inversa del desplazamiento es mucho mas complicado y se pierde informacion.

**EVOLUCIÓN DE LA SOLUCIÓN Y CONSIDERACIONES PARA FUTURAS IMPLEMENTACIONES**

**CONSIDERACIONES CLAVE APRENDIDAS:**

* Siempre validar todos los valores de límite y rango del arreglo para evitar errores de la memoria.
* Utilice funciones que combinen la lógica de verificación y aplicación para reducir el exceso.
* La liberación de la memoria debe ser sistemática para evitar fugas.
* La documentación impresa de cada función garantiza que el código sea más mantenible.
* Utilizar arreglos lineales simples facilita la manipulación a nivel de bits.