****

**Estudiantes:**

**Manuela Arboleda Montoya**

**CC: 1040571851**

**Juan Jose Medina Mejia**

**CC: 1036676459**

**Institución:**

**Universidad de Antioquia**

**Desafío 1**

**Curso:**

**Informática 2**

**Solución para la reconstrucción de imágenes a partir de datos de Transformación.**

El Desafío del Bit: Reconstruyendo la Imagen Perdida.

**Introducción**

Este informe detalla el enfoque para resolver el Desafío 1, que consiste en la construcción de una imagen BMP que ha sido sometida a una serie de transformaciones a nivel de bits. Se comprende la manipulación de datos, y es fundamental, ya que nos permite diseñar sistemas robustos, y también desarrollar habilidades para analizar y revertir procesos.

La solución se implementará en C++ utilizando punteros y memoria dinámica, lo que proporcionará una perspectiva práctica de la arquitectura de los sistemas computacionales. Tal como se especifica en las instrucciones, no se emplearán las estructuras de datos de la STL.

**Análisis del problema**

Este análisis implica una comprensión profunda de reconstruir una imagen BMP a partir de una serie de transformaciones bit a bit. Este proceso, requiere desglosar las transformaciones individuales y diseñar estrategias para revertirlas, considerando el orden inverso en el que se aplicaron.

Los archivos proporcionados para abordar este desafío incluyen una imagen final alterada (ID), una imagen intermedia (IM) necesaria para la operación XOR, una máscara (M) y archivos .txt. Estos archivos contienen la suma de los valores entre la imagen original desplazada y la máscara.

Los aspectos clave para resolver el desafío serían:

1. Estructura de las imágenes BMP: Comprender cómo se organizan los datos de los píxeles en un archivo BMP, incluyendo la representación de los colores RGB.
2. Operaciones a nivel de bits: Análisis de los operadores lógicos, y su efecto en los píxeles y las estrategias para revertirlas.
3. Proceso de enmascaramiento: Aplicación de la máscara, influencia del desplazamiento en los píxeles seleccionados y relación con los datos de los archivos .txt.
4. Relación entre los archivos de entrada: Establecer la conexión entre la imagen alterada (ID), la imagen intermedia (IM), la máscara (M) y los archivos .txt, y cómo cada uno contribuye al proceso de reconstrucción.

**Enfoque de solución**

El enfoque de esta solución se centra en la reconstrucción de la imagen original mediante la revisión sistemática de las transformaciones aplicadas, siguiendo las siguientes etapas clave:

1. Manejo de imágenes: Se implementan funciones para cargar BMP en memoria usando punteros y archivos binarios
2. Reversión de operaciones: Se desarrollarán funciones para aplicar e invertir operaciones como XOR,rotaciones y desplazamientos de bits.
3. Simulación de enmascaramiento: Crear una función que aplique la máscara y calcule las sumas **S(k) = ID(k+s) + M(k)** para la comparación con los archivos de entrada.
4. Lectura de datos de enmascaramiento: Leer los archivos .txt para extraer el desplazamiento y la secuencia de sumas resultantes del enmascaramiento.
5. Validación de las secuencias de las transformaciones: Comparar las sumas calculadas con los datos de los archivos .txt para verificar la correcta secuencia de transformaciones aplicadas.
6. Reconstrucción de la imagen: Una vez identificadas las transformaciones, aplicarlas en orden inverso para reconstruir la imagen original paso a paso.

**Plan de implementación**

Para resolver el Desafío I, se planteará una solución modular en C++ que permita abordar cada parte del problema de manera clara y ordenada. El proyecto se dividirá en componentes, cada uno con una responsabilidad específica, lo que facilitará tanto su comprensión como su mantenimiento.

Primero, se desarrollará un módulo llamado BmpImage.cpp. Este se encargará de leer archivos BMP monocromáticos, interpretar su estructura, extraer la matriz de bits correspondiente a la imagen y, posteriormente, permitir guardar nuevas imágenes modificadas. El objetivo es que toda la interacción con los archivos de imagen se maneje desde este módulo.

Luego, se implementará BitMask.cpp, que incluirá funciones para aplicar operaciones lógicas bit a bit entre imágenes: AND, OR, XOR y NOT. Estas funciones toman como entrada dos matrices de bits (o una, en el caso de NOT) y devolverán una nueva matriz resultante.

Una vez estén listos estos módulos, se escribirá el archivo main.cpp, desde el cual se gestionará el flujo principal del programa. En este archivo se cargarán las imágenes de entrada, se aplicarán las operaciones bit a bit, y se guardarán los resultados como nuevas imágenes BMP. De esta manera, será posible visualizar gráficamente los efectos de cada operación lógica sobre los bits de las imágenes.

Este enfoque modular permitirá reutilizar el trabajo hecho para futuros casos del desafío, enfocándose únicamente en la lógica específica que se requiera en cada parte, sin tener que modificar lo ya implementado.