Informe de análisis y diseño - Desafío 1: Reconstrucción de Imagen BMP

# 1. Introducción

Este informe presenta el análisis y diseño de una solución al Desafío 1. El problema planteado consiste en reconstruir una imagen BMP original que ha sido modificada mediante transformaciones a nivel de bits y técnicas de enmascaramiento, sin conocer el orden exacto de dichas transformaciones.

# 2. Análisis del Problema

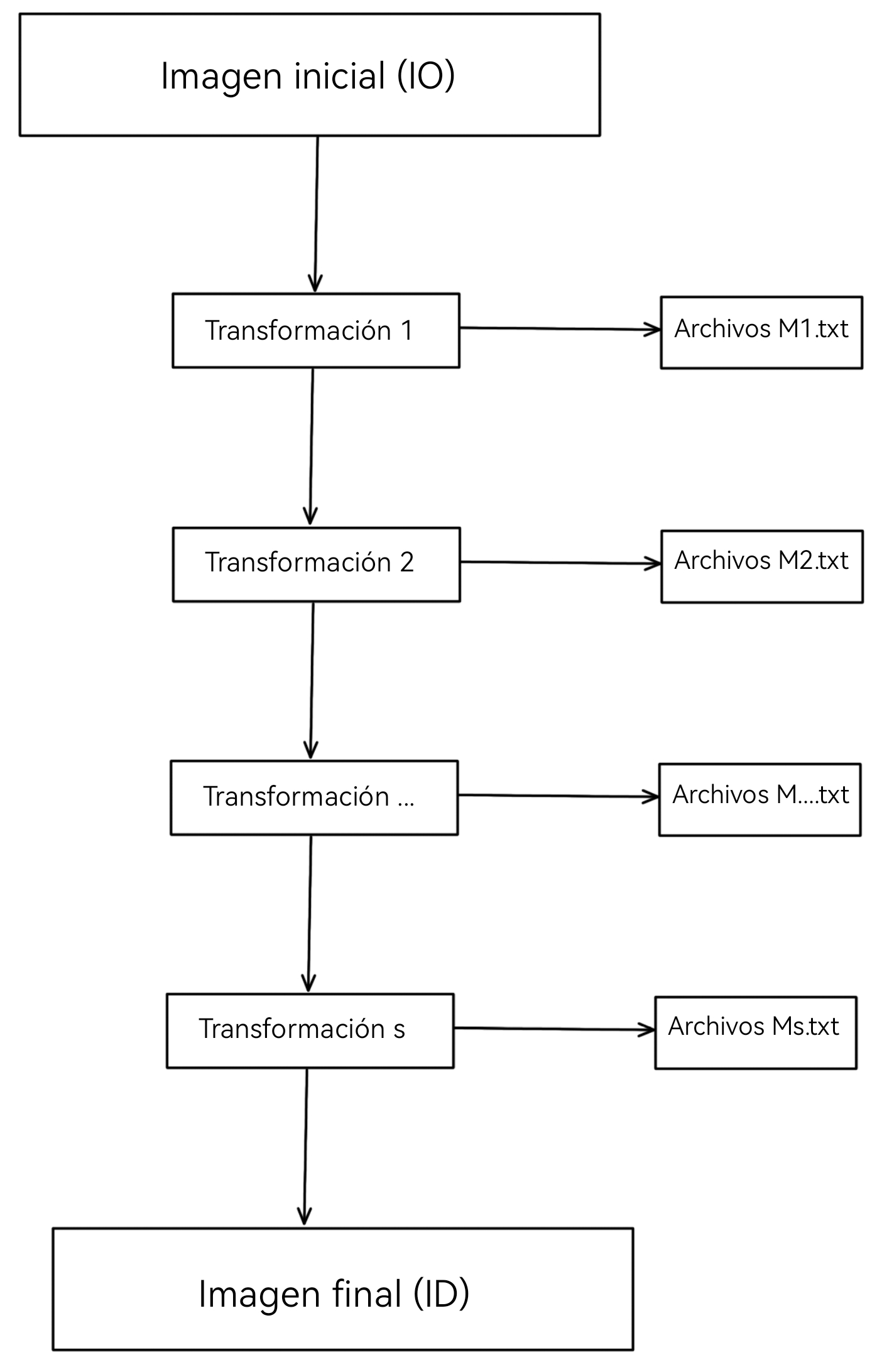
La imagen original (I\_O) fue transformada mediante una serie de operaciones a nivel de bits, entre las que se incluyen:  
  
- Operación XOR con una imagen aleatoria (I\_M).  
- Rotaciones de bits (por ejemplo, rotación derecha) y desplazamientos (shift) de bits, siendo el máximo de bits a manipular 8.  
- Enmascaramiento con una máscara M a partir de un desplazamiento aleatorio (semilla s).  
  
Cada transformación, en algunos casos, se acompaña del enmascaramiento, cuyos resultados se registran en archivos de texto, proporcionando así las únicas pistas para deducir el orden y los parámetros de cada operación. El objetivo es, a través de un proceso de ingeniería inversa, identificar las transformaciones aplicadas y revertirlas para obtener la imagen original.

# 3. Objetivo

El principal objetivo es diseñar e implementar en C++ (utilizando el framework Qt) un sistema que reconozca la secuencia de transformaciones utilizadas y, mediante la aplicación de la inversa de cada operación en el orden correcto, reconstruya la imagen original I\_O. Es importante notar que el reto implica un manejo eficiente de memoria dinámica, uso de punteros, y la implementación manual de operaciones a nivel de bits.

# 4. Diseño de la Solución

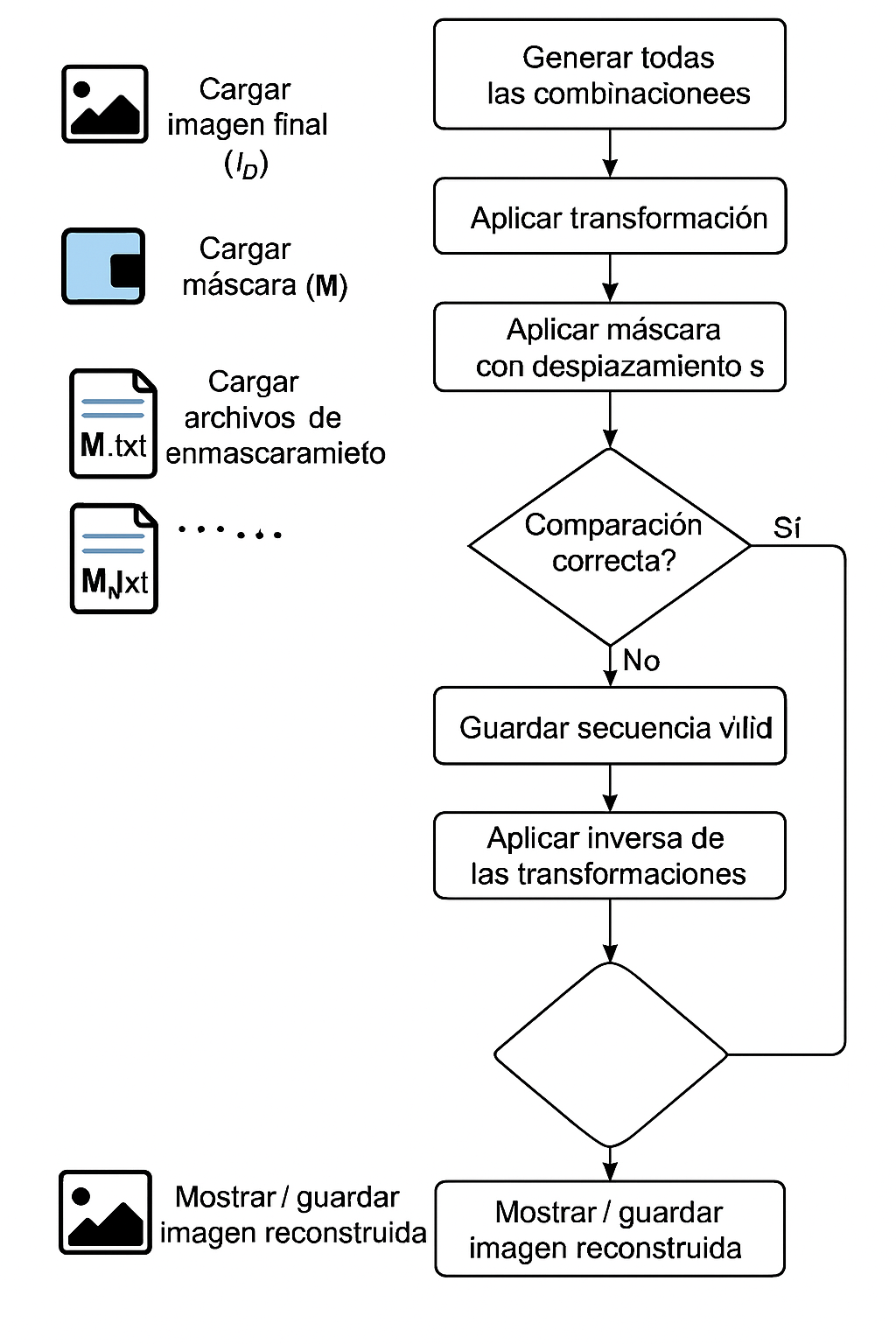
## 4.1 Modelo del Flujo de Datos

El proceso se puede representar de la siguiente forma:  
  
  
  
Las operaciones de transformación pueden ser de tres tipos:   
- XOR con I\_M.  
- Rotación de bits.  
- Desplazamiento lógico de bits.  
  
Cada enmascaramiento consiste en sumar pixel a pixel los valores RGB de la imagen transformada (a partir de un cierto desplazamiento 's') y la máscara M.

## 4.2 Funcionalidades y Módulos Propuestos

El sistema se divide en los siguientes módulos:  
  
1. Carga y exportación de imágenes:   
 - Funciones para leer archivos BMP y extraer los datos RGB en un arreglo dinámico.  
 - Funciones para exportar una imagen a BMP a partir de un arreglo de píxeles.  
  
2. Transformaciones a nivel de bits:   
 - Funciones para realizar XOR entre imágenes.  
 - Funciones para rotar bits (rotación a la derecha/izquierda) y para desplazar bits.  
  
3. Verificación del enmascaramiento:   
 - Función que toma un segmento de la imagen transformada a partir de un desplazamiento s, le suma la máscara M, y compara los resultados con los datos almacenados en los archivos .txt.  
  
4. Reconstrucción mediante búsqueda:   
 - Implementación de un algoritmo pruebe múltiples combinaciones de transformaciones y valide, a través del enmascaramiento, cuál es el orden correcto.  
 - Una vez identificada la secuencia, se aplica el proceso inverso para recuperar I\_O.

## 4.3 Esquema de Flujo



## 5. Código Base Proporcionado y Comentarios

Como punto de partida, se proporciona un código en C++ que realiza las siguientes tareas:  
  
- Carga una imagen BMP y almacena sus datos en un arreglo dinámico.  
- Aplica una modificación artificial a la imagen (degradado en base a la posición).  
- Exporta la imagen modificada a un nuevo archivo BMP.  
- Lee un archivo de texto que contiene una semilla y valores RGB resultantes del enmascaramiento.  
  
Este código sirve de base para implementar las funciones adicionales necesarias (XOR, rotación, desplazamiento y verificación del enmascaramiento) y se integrará en el módulo de reconstrucción mediante prueba y error de combinaciones de transformaciones.

## 6. Plan de Desarrollo

El plan de desarrollo propuesto es el siguiente:  
  
Semana 1:  
- Análisis del problema y familiarización con el código base.  
- Pruebas con imágenes BMP y manejo de archivos.  
  
Semana 2:  
- Implementación de funciones de transformaciones a nivel de bits (XOR, rotación y desplazamiento).  
- Pruebas unitarias de cada función.  
  
Semana 3:  
- Integración de la verificación de enmascaramiento y desarrollo del algoritmo de búsqueda.  
- Registro y análisis de resultados.  
  
Semana 4:  
- Documentación final, grabación del video de presentación y preparación de la sustentación.