**Analisi Desafío 1.**

**Elaborado por:**

* Maria Alejandra Gutierrez Rengifo
* Juan Camilo Perez

**Asignatura:** Informatica 2

**Docente:** Augusto Enrique Salazar Giménez

**Universidad de Antioquia**

**Facultad de Ingeniería**

**Programa de Ingeniería de Telecomunicaciones Medellín, 2025-1**

**Análisis del problema**

El presente proyecto surge como una tarea de ingeniería inversa aplicada a la recuperación de una imagen BMP de 24 bits que ha sido sometida a una serie de transformaciones desconocidas a nivel de bits. Las operaciones que pudieron haber sido aplicadas incluyen desplazamientos lógicos (shifts), rotaciones circulares (rotates) y operaciones XOR con una imagen de distorsión. Cada una de estas transformaciones fue seguida de un enmascaramiento parcial mediante una máscara de color, cuyos resultados fueron almacenados en archivos .txt. Estos archivos contienen una semilla (posición de inicio) y un vector de sumas RGB entre una sección de la imagen transformada y la máscara.

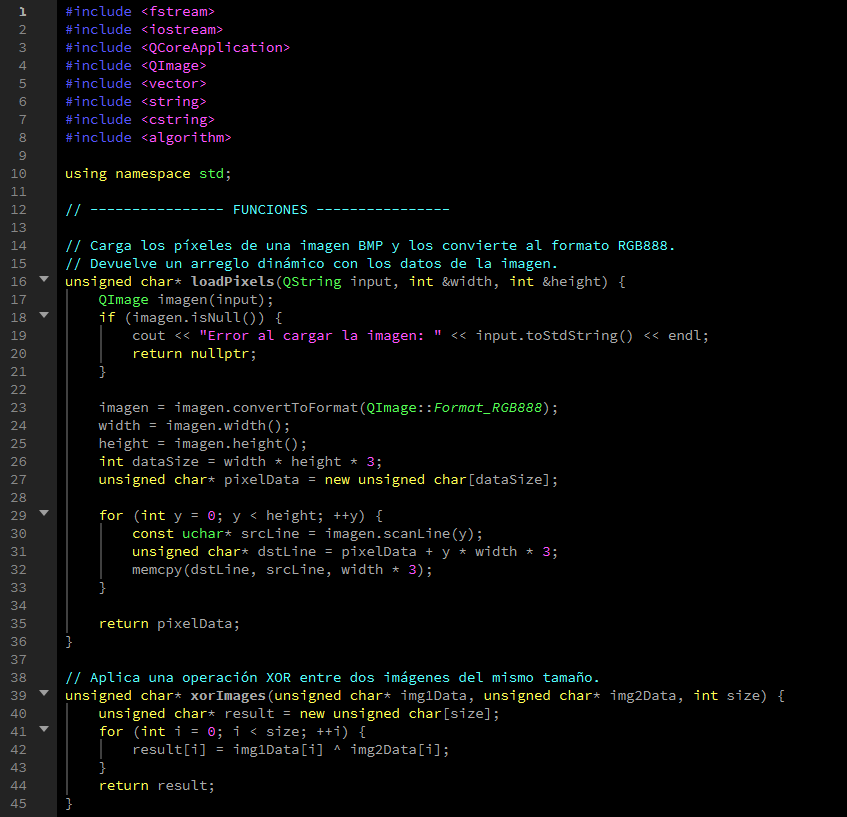
La dificultad principal del problema radica en que el orden y tipo de transformaciones aplicadas es completamente desconocido. Sin embargo, los archivos de sumas enmascaradas permiten verificar si una transformación particular genera el resultado esperado, por lo que una solución es posible mediante prueba y validación sistemática de operaciones.

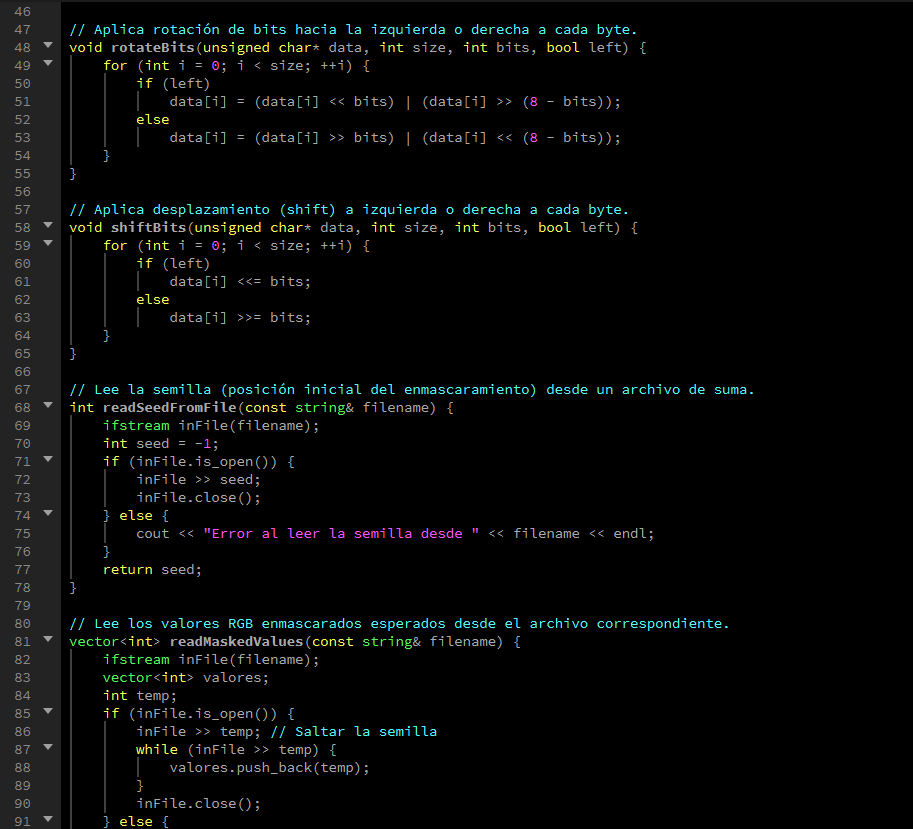
La solución propuesta se basa en el desarrollo de un sistema automatizado en lenguaje C++ con Qt, capaz de simular cada posible transformación y validarla contra los datos de los archivos de enmascaramiento. Al detectar coincidencias, el sistema deduce la operación correcta aplicada en esa etapa y reconstruye la imagen original paso a paso.

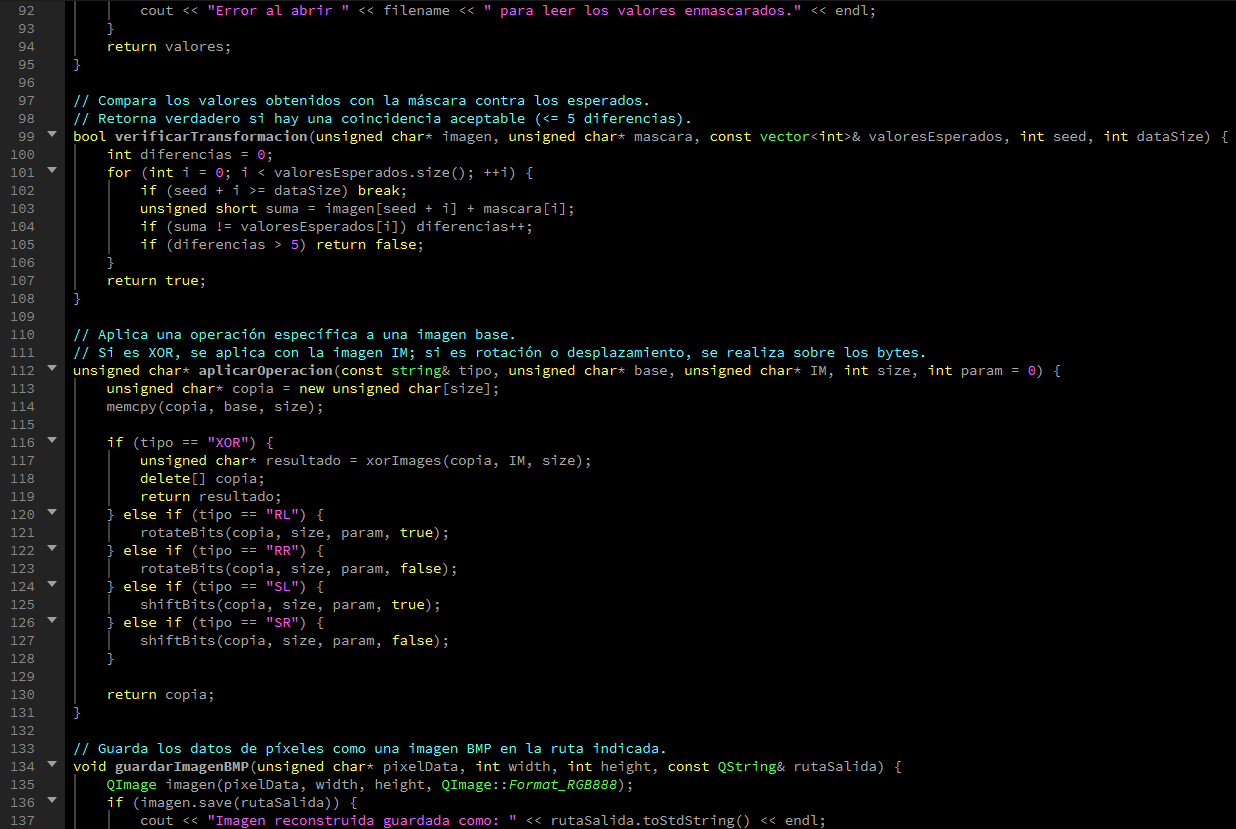
**Esquema de tareas definidas en el desarrollo de los algoritmos**

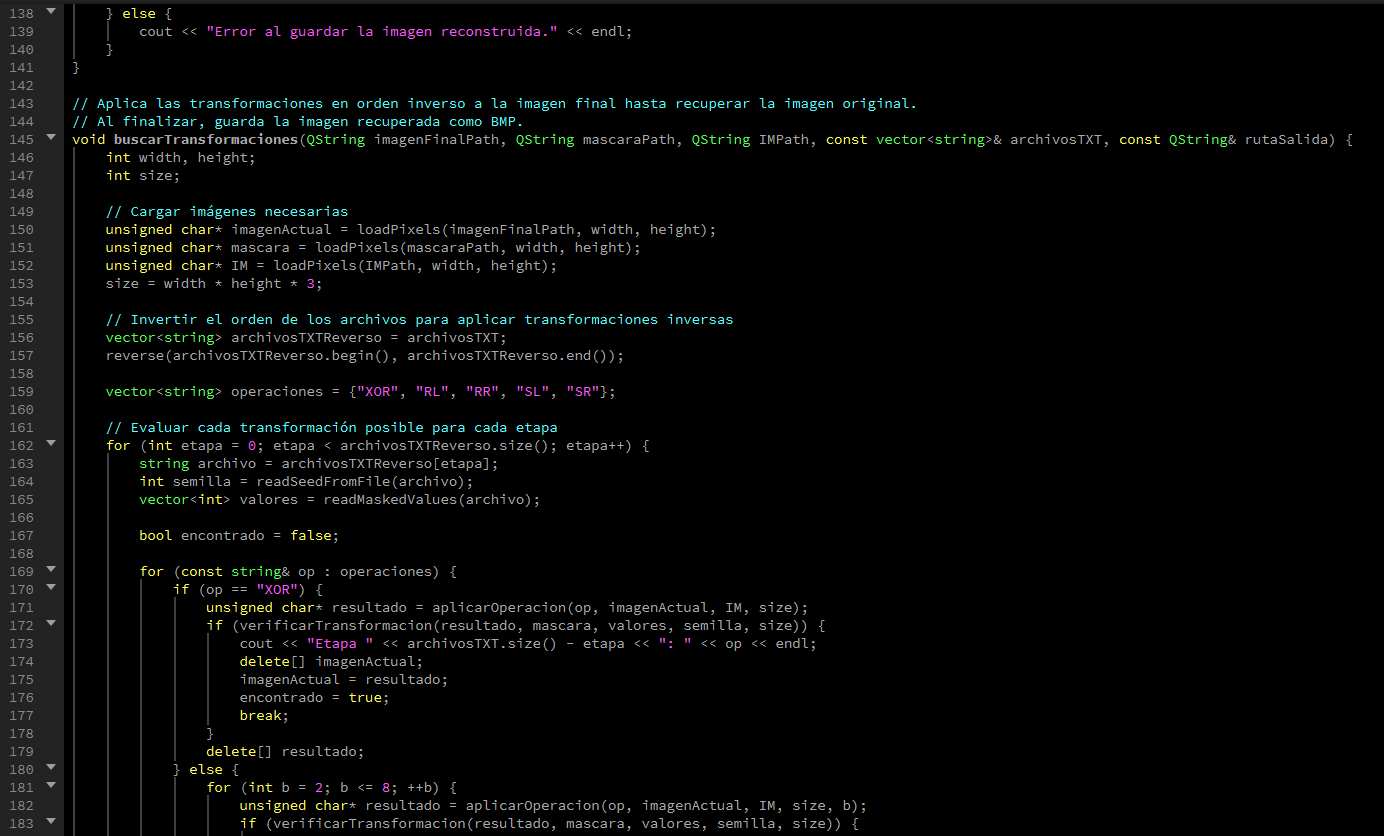
1. Lectura y procesamiento de imágenes BMP
   * Cargar imagen transformada (final).
   * Cargar máscara (M).
   * Cargar imagen de distorsión (IM).
2. Lectura de archivos de enmascaramiento
   * Leer semilla (posición de inicio).
   * Leer valores RGB esperados.
3. Definición de transformaciones
   * XOR con IM.
   * Rotación de 2 a 8 bits (izquierda y derecha).
   * Desplazamiento de 2 a 8 bits (izquierda y derecha).
4. Validación de transformaciones
   * Aplicar cada transformación posible.
   * Verificar si la suma entre sección transformada y máscara coincide con valores del archivo.
5. Registro y aplicación de transformaciones correctas
   * Una vez identificada la transformación correcta de una etapa, actualizar la imagen actual.
6. Guardado de la imagen reconstruida final
   * Exportar como archivo BMP la imagen recuperada.

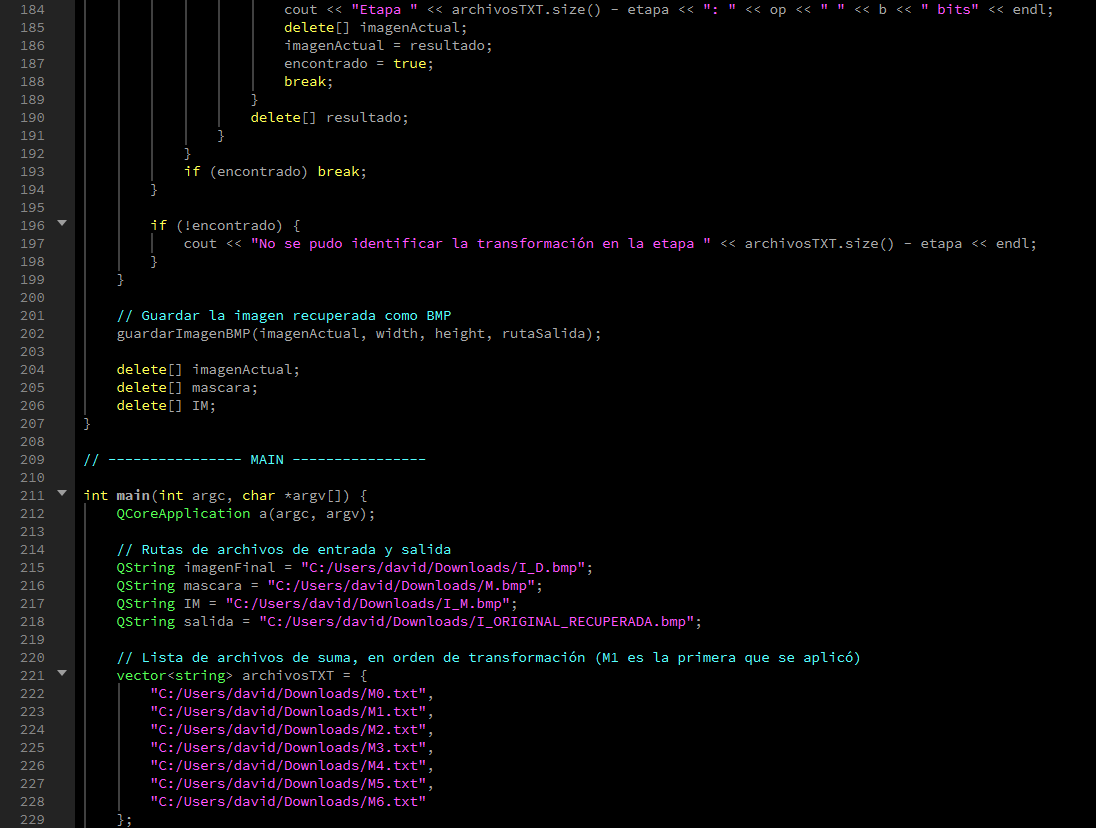
**Algoritmos implementados**

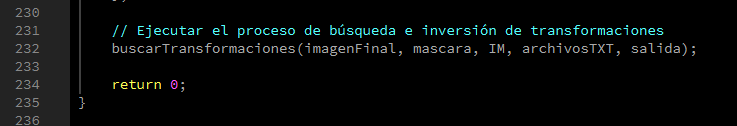












**Problemas de desarrollo que se afrontaron**

El uso de reverse() sobre un const vector& provocó errores de compilación que se resolvieron al crear una copia modificable del vector.

Fue necesario validar límites de acceso para evitar lecturas fuera de rango.

Al principio, el código evaluaba transformaciones con datos incorrectos (semilla fija). Luego se automatizó la lectura y se parametrizó.

Inicialmente no se visualizaba el efecto de cada transformación; más adelante se planteó exportar resultados intermedios para depuración.

**Evolución de la solución y consideraciones para la implementación**

Inicialmente se construyó una versión estática del programa que evaluaba manualmente una sola transformación por ejecución. Luego se generalizó a un sistema de prueba automática de operaciones para cada etapa.

Se adoptó una estrategia modular, separando la carga de datos, aplicación de operaciones y validación de resultados, lo que mejoró la legibilidad y mantenibilidad del código.

Finalmente, se integró la exportación de la imagen recuperada como BMP, lo cual marcó el cierre completo del flujo de reconstrucción.

**Recomendaciones para futuras mejoras:**

* Implementar interfaz gráfica para selección de archivos.
* Exportar imágenes intermedias por etapa.
* Incluir validación visual o de hash para comparar con la imagen original.
* Mejorar eficiencia usando paralelismo para aplicar operaciones simultáneas.

Este proyecto permitió aplicar conocimientos de manipulación de bits, manejo de memoria, lectura de archivos y programación modular en C++ con Qt, en un escenario realista de reconstrucción de información.