Nombre: Carlos Mario Rendón Martínez
Curso: Informática II
Semestre: 2025 – 1
Desafío: Reconstrucción de imagen BMP con transformaciones a nivel de bit

A. Análisis del problema y consideraciones para la alternativa de solución propuesta
.
R//: El reto consiste en recuperar una imagen BMP original que fue transformada varias veces usando operaciones a nivel de bit como XOR, rotación y desplazamientos. Después de cada transformación, se aplicó un enmascaramiento con una máscara de color y una posición de inicio (semilla).

La dificultad principal es que no sabemos el orden en que se hicieron las transformaciones, ni cuál fue exactamente cada una, pero sí tenemos pistas: los archivos .txt que se generaron después de cada paso.

Mi propuesta fue dividir el problema en varias partes pequeñas:

1. Leer las imágenes y los archivos .txt.

2. Aplicar operaciones como XOR y rotación con punteros.

3. Probar combinaciones posibles para encontrar el orden correcto de transformaciones.

4. Verificar si los resultados coinciden con los archivos .txt.

Como todavía estoy aprendiendo, traté de usar lo que hemos visto en clase: punteros, arreglos dinámicos, operaciones con bits, y todo sin usar estructuras ni STL.

B. Esquema de tareas definidas en el desarrollo de los algoritmos

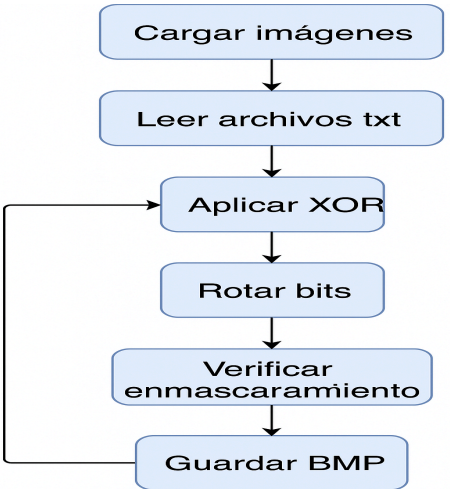
R//: 1) Cargar las imágenes (ID, IM y M) en memoria usando arreglos dinámicos y punteros.

2) Leer archivos .txt y guardar la semilla y los valores RGB del enmascaramiento.

3) Crear funciones para:

- Hacer XOR entre imágenes.
- Rotar bits a la derecha o izquierda.
- Desplazar bits.
- Aplicar el enmascaramiento y verificar si es correcto.

4) Probar diferentes órdenes de transformaciones y ver cuál logra reconstruir la imagen original.



5) Exportar la imagen final reconstruida en formato .bmp.

C. Algoritmos implementados.(Tabla de funciones principales)

Nombre de las funciones.	Entradas	Salidas	¿Qué hace?
leer Imagen BMP	Ruta del archivo BMP	Matriz RGB dinámica.	Carga una imagen en memoria, separando los valores RGB de cada píxel.
leer Archivo TXT	Ruta del archivo. TxT	Semilla y arreglo	Lee la semilla y los valores de enmascaramiento necesarios para el proceso.
aplicar XOR	Dos matrices RGB	Nueva matriz RGB	Aplique una operación XOR entre dos imágenes para obtener una imagen intermedia.
rotar Bits Derecha	Valor de color(int), cantidad de bits.	Valor rotado (int)	Rota los bits del color hacia la derecha para alterar su representación.
desplazar Bits Izquierda	Valor de color (int), cantidad de bits	Valor desplazado (int)	Mueve los bits del color hacia la izquierda, cambiando el valor del color.
aplicar Enmascaramiento	Imagen transformada	Imagen enmascarada	Aplique una máscara sobre la imagen para verificar si se cumple la condición.
verificar Resultado	valores.txt	Booleano (verdadero/falso)	Compara los valores de la imagen enmascarada con los que vienen en el .txt.
guardar Imagen	Matriz RGB	Archivo BMP	Guarda la imagen

BMP			reconstruida en un nuevo archivo BMP.

D. Problemas de desarrollo que afronté.

Aunque todavía no tengo los algoritmos completos, sí he estado investigando y tratando de entender cómo debo hacer cada parte del proyecto. En ese proceso me he encontrado con varias dificultades:

- Manejo de memoria dinámica:
  Me costó entender cómo se usan los punteros con arreglos dinámicos, especialmente cuando se trata de imágenes RGB, que usan punteros triples (***). A veces me enredaba al acceder a los datos o me olvidaba de liberar la memoria con delete, lo cual puede generar errores más adelante.

- Entender el formato BMP y el orden de los colores:
  Me di cuenta de que en las imágenes BMP, los píxeles están organizados por bytes y cada color tiene un orden específico. Esto fue confuso al principio, así que busqué ejemplos binarios y explicaciones para poder visualizar cómo se almacena cada canal (R, G y B).

- Transformaciones y su orden desconocido:
  El reto más grande ha sido que no sabemos en qué orden se aplicaron las transformaciones (XOR, rotación, etc.). Eso significa que hay que probar diferentes combinaciones, pero sin tener claro por dónde empezar, se vuelve algo frustrante.

- Errores de lógica y desbordamiento:
  Aunque no tengo el programa funcionando, en las pruebas que hice con posiciones de memoria y semillas, me di cuenta de que si la semilla s se usa mal, puede acceder a una parte de memoria que no corresponde, y eso puede hacer que el programa se caiga.

- Dificultad para empezar a escribir el código completo:
  A veces me pasa que entiendo la lógica general de lo que quiero hacer, pero me cuesta mucho empezar a escribirlo en C + +, sobre todo sin usar estructuras ni vectores (porque están prohibidos en este proyecto).

E. Evolución de la solución y consideraciones para la implementación
R//: Aunque todavía no he llegado a tener una solución completa, mi proceso ha ido avanzando poco a poco. Al principio me sentí muy perdido, especialmente porque este proyecto involucra muchas cosas al mismo tiempo: imágenes, bits, punteros, archivos y memoria dinámica.

Lo primero que hice fue tratar de entender bien el problema leyendo el PDF y haciendo apuntes. Me concentré en entender qué es lo que hace cada transformación (XOR, rotación, enmascaramiento) y cómo afectan la imagen. Luego empecé a investigar cómo se maneja una imagen BMP en C + + y cómo se puede representar en arreglos con punteros.

Al ir viendo más ejemplos y preguntando, entendí que necesitaba dividir el proyecto en partes pequeñas para no abrumarme: leer imágenes, hacer operaciones bit a bit, aplicar una máscara, y así. Aunque todavía no he logrado implementarlas todas, ya tengo una idea más clara de qué necesito hacer.

Algo importante que aprendí es que en este tipo de problemas es mejor probar cada parte por separado, antes de intentar hacer todo junto. Por ejemplo, probar que el XOR funciona bien entre dos valores antes de hacerlo con imágenes completas.
También aprendí que es fundamental controlar la memoria que uno reserva, porque si se olvida liberar con delete, el programa puede llenarse de errores que son difíciles de encontrar.