**Informe-Proyecto**

**Alumno: Augusto Enrique Salazar Jimenez**

**Docente: Rafael Nuñez Diaz**

****

**Informática II Teoría (2025-1)**

**Caucasia, Antioquia**

**2025**

**1.Análisis del problema.**

**Este proyecto consistía en recuperar una imagen BMP que fue modificada a nivel de bits mediante operaciones como rotación, desplazamiento y XOR. Para saber si la imagen recuperada era la correcta, se nos dio una máscara (M.bmp) y un archivo de verificación (M1.txt) que contiene una semilla (posición de inicio) y los valores RGB esperados tras aplicar esa máscara.**

**La dificultad principal del problema era que no sabíamos qué operaciones exactas se usaron ni en qué orden, por lo que la solución debía probar diferentes combinaciones hasta encontrar una que generara los valores correctos enmascarados.**

**2. Esquema de trabajo**

**Estas fueron las tareas que fui definiendo y desarrollando durante el proceso:**

1. **Leer las imágenes BMP (I\_O.bmp y I\_M.bmp) usando Qt.**
2. **Leer el archivo M1.txt, que contiene la semilla y los valores RGB a verificar.**
3. **Cargar la máscara M.bmp, la cual es usada para validar si una imagen fue correctamente reconstruida.**
4. **Aplicar rotaciones y XOR en diferentes órdenes para tratar de recuperar la imagen original.**
5. **Verificar si la imagen resultante coincide con los datos del archivo M1.txt.**
6. **Exportar la imagen recuperada si se logró encontrar la secuencia correcta.**
7. **Imprimir por consola el resultado del proceso, tanto los datos cargados como si se logró o no la verificación.**

**3. Algoritmos y funciones**

**Para resolver el desafío, usé las siguientes funciones:**

* **loadPixels(...): carga una imagen BMP y extrae sus valores RGB a un arreglo.**
* **exportImage(...): guarda un arreglo RGB como una imagen BMP.**
* **applyXOR(...): aplica la operación XOR entre dos arreglos de píxeles.**
* **rotateBitsLeft(...) y rotateBitsRight(...): hacen rotaciones circulares de bits.**
* **loadSeedMasking(...): lee la semilla y los valores enmascarados desde un archivo .txt.**
* **verificarEnmascaramiento(...): compara el resultado de aplicar la máscara con los valores esperados.**

**Además, incluí un ciclo para probar diferentes combinaciones de rotaciones, ya que no se sabía con certeza cuántos bits fueron rotados ni en qué orden.**

**4. Dificultades**

**Durante el desarrollo me enfrenté a varios retos:**

* **Al principio no usaba la máscara real (M.bmp), sino una simulada llena de ceros. Esto hacía que la verificación nunca funcionara.**
* **Había confusión sobre si el enmascaramiento se hizo por suma o por XOR, así que tuve que probar ambas opciones.**
* **Tuve que asegurarme de que el tamaño de la máscara coincidiera exactamente con la cantidad de píxeles leídos del archivo.**
* **La lectura del archivo .txt requería pasarlo dos veces: una para contar los píxeles y otra para cargar los datos correctamente.**
* **También me costó entender el orden exacto en que aplicar las operaciones para revertir las transformaciones.**

**5. Cómo evolucionó la solución**

**Comencé con un programa muy básico que solo leía una imagen y mostraba sus píxeles. Luego fui agregando la lectura de los archivos de verificación, probé XOR, rotaciones y al final integré la carga de la máscara real desde M.bmp.**

**Cuando logré hacer que el programa recorriera automáticamente las posibles rotaciones y verificara la imagen usando la máscara real, fue cuando el sistema empezó a dar resultados más precisos. También añadí mensajes en consola para entender en qué paso fallaba o si había coincidencia.**

**6. Conclusiones**

**Este proyecto me ayudó a practicar el manejo de imágenes a nivel de bits en C++, el uso de memoria dinámica, y sobre todo, a trabajar con procesos inversos donde no se conoce a priori el orden de las operaciones. También reforcé el uso de archivos, punteros y estructuras de bajo nivel sin depender de herramientas avanzadas.**

**Quedé satisfecho con el resultado, y creo que el programa puede extenderse fácilmente para manejar varios casos automáticamente o incluso integrarse con una interfaz gráfica.**

**7. Recomendaciones (por si alguien lo mejora después)**

* **Sería ideal automatizar más combinaciones: no solo rotación y XOR, sino también desplazamiento de bits.**
* **Agregar verificación con M2.txt (si hay más máscaras) para pruebas dobles.**
* **Incluir una interfaz visual donde se vean las imágenes procesadas y sus máscaras.**
* **Validar que la semilla nunca cause lecturas fuera del rango del arreglo de imagen.**