Estructura del Programa – Desafío 1 Informática II (UdeA)

Este documento describe la estructura lógica y técnica del programa desarrollado para resolver el Desafío 1 de la asignatura Informática II, en el marco del semestre 2025-1 del programa de Ingeniería Electrónica de la Universidad de Antioquia. Se detallan las decisiones de diseño, fases del algoritmo, optimizaciones de memoria, así como las variables utilizadas.

Trabajo realizado por:  
Juan Paulo Moncada Morales - CC 1059699215  
LÓPEZ RAMÍREZ EMMANUEL - CC 1038866909  
Grupo 3

# 1. Estructura general del programa (Resumen)

## 1. Carga de datos a memoria dinámica

Se cargan en memoria dinámica: la imagen final transformada (ID), la imagen auxiliar (IM), la máscara (M) y todos los archivos .txt. Se verifica la validez de dimensiones entre imágenes y máscara. Se cumple con la restricción de no usar STL ni estructuras.

## 2. Preprocesamiento durante la carga: desenmascaramiento inmediato

Durante la carga de los archivos .txt se realiza la resta directa de la máscara M a los valores RGB, sobrescribiendo estos valores en el mismo arreglo. Esto permite liberar la memoria de la máscara inmediatamente después del preprocesamiento, ya que no se vuelve a usar.

## 3. Estrategia de desencriptado con doble buffer

Se utilizan dos arreglos dinámicos (bufferA y bufferB) para aplicar operaciones inversas sin sobrescribir datos críticos. Los buffers alternan sus roles a través de punteros (actual, siguiente) que se intercambian en cada iteración.

## 4. Bucle principal de desencriptado

Se recorren los archivos .txt en orden inverso. En cada paso se prueban todas las combinaciones posibles de operaciones inversas (XOR, rotaciones, desplazamientos) con diferentes parámetros. La verificación se realiza con los datos desenmascarados previamente.

## 5. Verificación con fragmentos desenmascarados

Cada operación candidata se verifica comparando directamente los bytes de la imagen generada con los valores almacenados en el .txt correspondiente. Si coinciden, se registra la operación y se intercambian los buffers. La memoria del .txt verificado se libera.

## 6. Finalización

Al completar todos los pasos, el arreglo de operaciones registradas contendrá la secuencia inversa completa. El último buffer activo contendrá la imagen reconstruida. Se puede exportar en formato BMP para validación visual.

## 7. Consideraciones clave

El programa optimiza el uso de memoria liberando estructuras no necesarias como la máscara M y evitando estructuras auxiliares como semillas y fragmentos preprocesados. Toda la lógica se basa en punteros y arreglos dinámicos, cumpliendo con las restricciones del desafío.

# 2. Tabla de Variables y Arreglos Utilizados

| Nombre | Tipo de dato | Descripción | Uso en el código |
| --- | --- | --- | --- |
| bufferA | unsigned char\* | Contiene la imagen inicial (ID) y alterna con bufferB | Sirve como imagen actual para operaciones inversas |
| bufferB | unsigned char\* | Segundo arreglo para almacenar el resultado de operaciones | Permite alternar sin sobrescribir bufferA |
| actual, siguiente | unsigned char\* | Punteros auxiliares | Controlan la alternancia entre los buffers |
| txt\_data[k] | int\* | Arreglo dinámico por archivo .txt, sobrescribe los valores tras desenmascarar | Se compara en cada paso con la imagen generada |
| operaciones[k] | int\* | Arreglo dinámico con tipo y parámetro de operación aplicada | Registra la secuencia de operaciones inversas |
| tamaño\_img | int | Total de bytes de la imagen (m × n × 3) | Para inicialización de buffers y bucles de recorrido |
| semilla\_actual | int | Semilla actual extraída del .txt en curso | Se usa solo en la etapa activa |
| ancho\_mask, alto\_mask | int | Dimensiones de la máscara | Se usan para calcular cuántos bytes comparar |
| coincide | bool | Bandera de verificación | Indica si la operación aplicada fue válida |
| k, i, j | int | Variables auxiliares | Controlan los ciclos y recorridos de datos |