Desafío 1 – Ingeniería inversa de un mensaje comprimido y encriptado

Curso: Informatica II

Integrantes:

* Brandon Manuelle Morales Cardona
* Juan Camilo Ramirez Lozano

Grupo: 03

Fecha de entrega: 19 de septiembre de 2025

# Introducción

El objetivo de este trabajo es aplicar conceptos de compresión de datos (RLE y LZ78) y criptografía básica (rotación de bits y XOR) para recuperar un mensaje original a partir de una versión comprimida y encriptada. Además, se busca implementar la solución en C++ utilizando punteros y arreglos dinámicos, sin apoyo de librerías STL.

# Análisis y diseño de la solución

El problema se descompone en cuatro módulos principales, lo que facilita la organización del código y la asignación de responsabilidades entre los integrantes del equipo:

- Módulo de desencriptación: encargado de revertir la encriptación bit a bit mediante rotaciones y operación XOR. Este módulo prueba diferentes valores de n (rotación) y K (clave XOR) hasta encontrar el fragmento conocido del mensaje original.  
- Módulo de descompresión RLE: implementa la lógica de Run-Length Encoding para expandir cadenas del tipo '4A3B2C1D2A' en 'AAAABBBCCDAA'.  
- Módulo de descompresión LZ78: gestiona un diccionario dinámico con punteros para reconstruir el mensaje a partir de pares (índice, carácter).  
- Módulo de integración: une todo el flujo. Recibe el mensaje encriptado, lo desencripta, prueba la descompresión con RLE y LZ78 y selecciona la que contenga el fragmento conocido.

Esta estrategia nos permitira trabajar de manera ordenada: un integrante se enfocará en la implementación de RLE y LZ78, mientras que el otro se encargará de la desencriptación. Ambos trabajaremos juntos en la integración final.

Plan de implementación propuesto:

1. Implementar y probar el módulo de RLE   
2. Implementar y probar el módulo de LZ78   
3. Implementar las funciones de rotación y XOR   
4. Probar desencriptación con valores de prueba y fragmento conocido  
5. Integrar los módulos en main.cpp

# Resultados esperados

Se espera que el programa sea capaz de:  
- Desencriptar el mensaje identificando los parámetros n y K.  
- Probar la descompresión con RLE y LZ78.  
- Determinar cuál método fue utilizado realmente en el mensaje recibido.  
- Recuperar el mensaje original de manera correcta.  
Además, se espera contar con un repositorio organizado y un informe que documente el proceso.

# Conclusiones

La estrategia de dividir la solución en módulos facilita tanto la implementación como la colaboración en equipo. Esta estructura asegura que cada integrante se enfoque en un aspecto claro del problema y luego se integren los avances. También se reconoce que la restricción de no usar STL obliga a profundizar en el manejo de punteros y memoria dinámica, fortaleciendo las habilidades de bajo nivel en C++.

# Referencias

- Material del curso  
- Documentación sobre compresión RLE y LZ78  
- Recursos de rotación de bits y XOR en C++