**Informe Preliminar Desafío 1 (Informática 2)**

**Estudiantes:**

Jose Eduardo Valverde Alvarez ([josee.valverdea@udea.edu.co](mailto:josee.valverdea@udea.edu.co))

Andrés Felipe Lafaurie Rincon ([andres.lafaurie@udea.edu.co](mailto:andres.lafaurie@udea.edu.co))

**Profesor:**

Anibal Jose Guerra Soler

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería en Telecomunicaciones e Ingeniería electrónica

Medellín

19 de Septiembre de 2025

**Contextualización:**

Un cliente nos entrega un mensaje original hecho en texto plano que pasó por dos tipos de codificación, primero pasó a ser codificado (o en este caso, comprimido) en uno de los dos siguientes formatos de compresión: RLE o Lempel-Ziv (LZ78), luego de pasar por el proceso de comprensión en uno de los dos formatos, el texto comprimido pasa a binario, dando lugar al proceso de rotación de entre 1 y 7 bits a la izquierda, y finalmente se le asignó una clave que fue proporcionada por el cliente la cual somete a un enmascaramiento usando el operador XOR a los bytes rotados, la clave ocupa el tamaño de un solo byte y está en hexadecimal, al hacer la conversión de cada uno de estos caracteres en binario rotados y enmascarados a char (ASCII/UTF-8) veremos lo que sería el conjunto de caracteres extraños en el archivo de texto que nos proporcionó el cliente.

**Análisis detallado del desafío:**

Analizamos el desafío según los requerimientos exigidos.

}

Sabemos que el mensaje fue comprimido y encriptado utilizando 2 posibles métodos para cada caso, en el caso de la compresión se pudo utilizar RLE o LZ, mientras que para la encriptación del mensaje se pudo utilizar rotación de bits y encriptado XOR.

Teniendo el mensaje encriptado y sabiendo que se utilizó alguno de los 2 métodos de compresión y después se aplicaron los métodos de rotación a la izquierda de bits junto con un enmascaramiento XOR una sola vez para pasar del mensaje original al encriptado analizamos la secuencia que tendríamos que seguir para llevar este hasta el punto que tenemos, deducimos entonces la secuencia realizada para la encriptación.

Texto original > Compresión > Transformación a binario > Aplicar la máscara de XOR > Rotación de bits a la izquierda > transformar de nuevo a ASCII > Texto encriptado

Conociendo este proceso sabemos que para desencriptar el texto debemos hacer el proceso inverso con el texto que tenemos encriptado y compararlo con la pista que nos dieron para comprobar que el proceso realizado sea el correcto.

Planteamos el modelo de código a seguir para realizar la desencriptación del mensaje.

El código planteado consiste en: dentro de la función Main se inicializa el programa, se le pide al usuario que ingrese el número de textos que desea descifrar, tras de esto se ingresa a un while que itere tantas veces como textos se desean descifrar, se procede a llamar a una función que permitirá crear nombres de los 2 archivos que se necesitan para empezar el descifrado, utilizando un contador que se aumentará con cada iteración del while, con los nombres de los archivos obtenidos y almacenados en una variable dinámica se llama a una función que lea y almacene el contenido de los archivos en variables dinámicas. Se va a llamar a una función que convertirá el texto encriptado en binario y lo almacenará en la misma variable, con el texto en binario se adentra en un ciclo que empezará el proceso de desencriptación .

El ciclo tendrá un contador que iniciará en 1 y se aumentará en 1 su valor por cada ciclo hasta llegar a 7.

La primera función del desencriptado es la rotación de bits, para después llamar una función que va aplicar la máscara necesaria para desencriptar el texto. Después de los pasos de encriptación el texto regresará a notación Ascii y se adentra en otro ciclo interno donde se probará descomprimir el texto que tenemos en los dos tipos de compresión posibles, al realizar cualquiera de las de impresiones por cualquiera de los métodos se comparará el texto resultante con la pista, en caso de que la pista se encuentre dentro de este texto quiere decir que el proceso realizado fue el correcto y se retorna el texto desencriptado junto con los datos de el número de bits rotados, la máscara usada y que método de de compresión se usó, en caso de que al realizar la comparación no fuera correcto el código el programa limpiará la memoria dinámica el proceso de desencriptado y las variables que se utilizaron para ese resultado y continuará el ciclo hasta encontrar la respuesta o recorrer todas las opciones posibles.

**Diseño:**

Teniendo en cuenta los análisis del problema,. Planteamos entonces las funciones que necesitaremos.

**Leer archivos.**

El primer paso es leer los archivos con la información del texto encriptado y la pista para obtener la información dentro de estos. Para ello desarrollaremos una función utilizando la librería fstream que lea el mensaje encriptado y lo guarde en un arreglo dinámico tipo char cuyo tamaño es variado según la cantidad de caracteres que tenga el archivo en ese momento, con el texto encriptado que se está leyendo para poder manipularlo y realizar el proceso para desencriptar. Este código también permitirá leer y almacenar la pista del mismo índice.

Para acceder a estos archivos crearemos una función que reciba el número de archivos que se van a evaluar, y lea los nombres que tienen estos para poder abrirlos usando la función anterior. Esto lo haremos con una función que lea los nombres de los archivos de texto que como patrón tengan el nombre “Encriptado” o “pista” al cual se le agregara el número del texto que se esté evaluando, este se conseguirá por medio de un ciclo y un contador desde 1 hasta el número de archivos que se van a evaluar para poder agregarlos a la base, y en caso de ser necesario agregar la extension “.txt”.

**Transformar char a binario**

El mensaje encriptado lo encontramos en forma de caracteres que resultan de la serie de encriptamientos a los bits a los que se sometieron, por lo que necesitamos primero una función que convierta el arreglo de caracteres copiado del archivo recibido a un conjunto de bits (con static\_cast y la librería bitset) que representa cada uno de los caracteres dentro de ese arreglo, esta función se hace caracter por caracter.

**Rotación de bits y XOR usando la clave.**

El primer paso para descifrar el mensaje es aplicar la rotación de bits a la izquierda, teniendo el mensaje transcrito a binarios creamos una función con un ciclo anidado que permita probar todas las posibilidades de bits rotados a la izquierda, este número estará entre 1 y 7 (0 < n < 8) por lo que el ciclo realizará las operaciones para cada una de estas opciones. Una vez seleccionado el número de bits a rotar, el programa modificará cada binario del texto rotando en cantidad de bits. Esto utilizando un ciclo que tome cada bit y le haga las operaciones necesarias.

Después de realizar la rotación de bits el siguiente paso es aplicar la misma máscara que se utilizó para encriptarla con el método XOR. Esta función utiliza un ciclo para extraer cada binario de cada carácter del texto, aplica la máscara a cada uno de los caracteres en binario del texto y escribiendo en el mismo arreglo, esperando obtener la representación binaria del texto comprimido.

**Descompresión**

Tras desencriptar el mensaje y convertirlo de vuelta a char, se debe descomprimir el mensaje utilizando alguno de los 2 posibles métodos RLE o LZ78.

* RLE: Se descompondrá el texto por secuencias, desde un número hasta otro, luego de cada secuencia se va a tomar el número, y se escribirá la letra que le sucede ese número de veces, el texto nuevo se escribirá en un nuevo arreglo.
* LZ78: Se descompone el texto en pares (indice, valor), si el índice está vacío agrega el valor a un diccionario y escribe el valor en el arreglo, en caso de no estar vacío busca dentro del diccionario el valor perteneciente a ese índice, lo escribe en el arreglo y le agrega el nuevo valor al final.

**Comparar texto desencriptado y pista**

Después de haber realizado todo el proceso de desencriptación y descompresión del mensaje, se debería tener el codigo original, en caso de que se hubieran aplicado los procesos correctos se debe comparar el texto con la pista y en caso en que la pista se encuentre dentro del texto procesado imprimir este y junto con variables donde se anotaron el número de bits rotados y la máscara usada. En caso de que la pista no se encuentre quiere decir que el proceso no fue el correcto y se borraran los datos almacenados y se comenzará de nuevo con otras ramas.