Informe de desarrollo del desafío 1

Estudiantes:

Kevin Esteban Echeverri Carmona 1000294820
Ivonne Lizeth Rosero Cardona 1007687589

a. Análisis del problema y consideraciones para la alternativa de solución propuesta

El problema planteado consiste en recuperar un mensaje en texto plano que ha sido sometido a un proceso de **compresión** (mediante los algoritmos RLE o LZ78) y posteriormente a un proceso de **encriptación** compuesto por dos transformaciones:

1. Rotación circular a la izquierda de cada byte en nnn posiciones.
2. Operación XOR con una clave KKK de un solo byte.

El único dato disponible es el mensaje comprimido y encriptado, junto con un fragmento conocido del mensaje original. Esto implica que el proceso de recuperación requiere no solo aplicar las operaciones inversas, sino también **identificar los parámetros exactos** del sistema: el método de compresión utilizado, el número de rotaciones nnn y la clave KKK.

Las principales consideraciones que se tuvieron en cuenta para definir una estrategia de solución son las siguientes:

- **Espacio de búsqueda reducido:** los valores posibles para la rotación (n∈[1,7]n \in [1,7]n∈[1,7]) y para la clave XOR (K∈[0,255]K \in [0,255]K∈[0,255]) generan un total de 4.096 combinaciones. Esto hace posible aplicar una búsqueda exhaustiva de parámetros sin comprometer la eficiencia.
- **Validación inequívoca:** la existencia de un fragmento conocido del texto original permite confirmar de manera determinista si una combinación candidata es correcta, evitando ambigüedades en la solución.
- **Estructura de los algoritmos de compresión:** tanto RLE como LZ78 presentan patrones y formatos específicos en la descompresión. Esto facilita descartar candidatos inválidos de manera temprana.
- **Modularidad:** la solución propuesta se estructura en módulos independientes para compresión/descompresión y encriptación/desencriptación, lo que facilita su validación y reutilización.

Con base en estas consideraciones, se optó por una solución **sistemática y exhaustiva**, que explora todas las configuraciones posibles, descarta rápidamente candidatos inválidos y valida los resultados a través del fragmento de texto conocido.

b. Esquema de tareas para el desarrollo de los algoritmos

El desarrollo del proyecto se estructuró en una serie de tareas, organizadas de manera secuencial y modular:

1. **Implementación de algoritmos de compresión y descompresión:**
    - Desarrollo de la compresión y descompresión mediante RLE.
    - Desarrollo de la compresión y descompresión mediante LZ78.
    - Validación de los algoritmos con ejemplos de prueba.
2. **Implementación de algoritmos de encriptación y desencriptación:**
    - Implementación de la rotación de bits hacia la izquierda y hacia la derecha.
    - Implementación de la operación XOR con clave de un byte.
    - Pruebas unitarias para verificar la correcta inversión de las operaciones.
3. **Diseño del módulo de búsqueda de parámetros (nnn y KKK):**
    - Generación de todas las combinaciones posibles de parámetros.
    - Aplicación de la desencriptación inversa a cada candidato.
4. **Pruebas de descompresión en candidatos desencriptados:**
    - Intento de descompresión con RLE.
    - Intento de descompresión con LZ78.
    - Descarte inmediato de secuencias no válidas.
5. **Validación con el fragmento conocido del mensaje:**
    - Comparación del resultado de la descompresión con el fragmento de texto de referencia.
    - Confirmación de la combinación correcta de parámetros.
6. **Reconstrucción final del mensaje original:**
    - Obtención íntegra del texto en claro.
    - Reporte del método de compresión, el valor de nnn y la clave XOR KKK.

c. Algoritmos implementados

Para el desarrollo del **Desafío N.° 1** se adoptará un esquema de **programación modular**, en el cual cada funcionalidad principal será implementada como un algoritmo independiente, lo que permitirá mantener un diseño claro, reutilizable y fácilmente verificable.

El sistema estará compuesto por **cinco módulos algorítmicos**, los cuales funcionarán como bloques de construcción y serán invocados por una función **principal (main)** encargada de coordinar el flujo de datos, la secuencia de operaciones y la validación de resultados.

- **Algoritmos RLE:**
    - *RLE Comprimir:* toma un flujo de texto y genera la representación comprimida bajo el esquema.
    - *RLE Descomprimir:* recibe un bloque comprimido en RLE y reconstruye el texto original.
- **Algoritmos LZ78:**
    - *LZ78_Comprimir:* implementa el método de compresión basado en diccionario dinámico (índice + símbolo).
    - *LZ78_Descomprimir:* procesa una secuencia de pares índice/símbolo y reconstruye el texto plano original.
- **Algoritmo de desencriptación:**
    - *Desencriptar Bloque:* recibe como parámetros el número de rotación de bits n y la clave XOR K, aplicando la operación inversa al proceso de encriptación para recuperar el flujo comprimido en su forma válida.

Con esta arquitectura, la **función principal** podrá combinar los módulos de acuerdo con el flujo definido: desencriptar → descomprimir → validar, o en sentido contrario, comprimir una frase conocida y compararla con la versión desencriptada. De este modo, se garantiza un desarrollo flexible y escalable, adecuado para las pruebas exhaustivas requeridas en el reto.

d. Problemas del desarrollo que afronto.

Durante el desarrollo del desafío, uno de los principales inconvenientes fue la **ambigüedad en la combinación de parámetros**: la clave de encriptación, el número de bits de rotación y el tipo de algoritmo de compresión utilizado. Esta multiplicidad de opciones genera un espacio de búsqueda considerable, lo cual representa un reto tanto en términos de **gestión eficiente de la memoria** como en el **uso de recursos de procesamiento**.

Otro problema relevante fue la aparición de **falsos positivos**. En algunos casos, ciertos parámetros producían resultados aparentemente correctos en un rango limitado de caracteres, pero que no correspondían realmente a la configuración válida del archivo. Este fenómeno podía inducir a errores de interpretación y desviar el flujo de trabajo. Para mitigarlo, se establecieron **mecanismos de validación adicionales**, como la verificación de coherencia global del texto descomprimido, el análisis de caracteres imprimibles y la comparación con la frase conocida provista en el enunciado.

e. Evolución de la solución y consideraciones para tener en cuenta en la implementación.

La evolución de la solución se plantea de manera gradual y estructurada. En una primera etapa se validará el correcto funcionamiento de cada uno de los algoritmos básicos —tanto los de compresión y descompresión (RLE y LZ78) como el de desencriptación mediante pruebas unitarias sobre fragmentos controlados.

Una vez verificada la robustez de estos módulos, se integrará un algoritmo de búsqueda exhaustiva, encargado de recorrer todas las combinaciones posibles de parámetros (rotación de bits, clave XOR y tipo de compresión). En esta fase, se contrastará el resultado con el fragmento conocido en sus dos formas: el original en texto plano y su contraparte encriptada y comprimida.

Cuando se identifiquen los parámetros correctos, se generará un listado de configuraciones candidatas, las cuales serán posteriormente evaluadas para determinar su veracidad y consistencia. Finalmente, la combinación válida se aplicará al archivo completo con el fin de recuperar el texto original.

En cuanto a las consideraciones para la implementación, se enfatiza especialmente en el uso adecuado de la memoria:

- Seleccionar los tipos de variables apropiados (uint8_t, size_t, etc.) para garantizar eficiencia en espacio y procesamiento.
- Gestionar correctamente los buffers dinámicos con new[]/delete[] o malloc/free para evitar fugas de memoria.
- Mantener una clara separación modular entre algoritmos, lo que facilita la depuración, la reutilización de código y el control de recursos.

Este enfoque asegura una evolución controlada del proyecto, garantizando tanto la validez funcional de los algoritmos como un uso óptimo de los recursos de hardware.

Diagrama de flujo