## **Desafío 1 - Informática II**

Juan José Rendón González  
C.C 1000415194

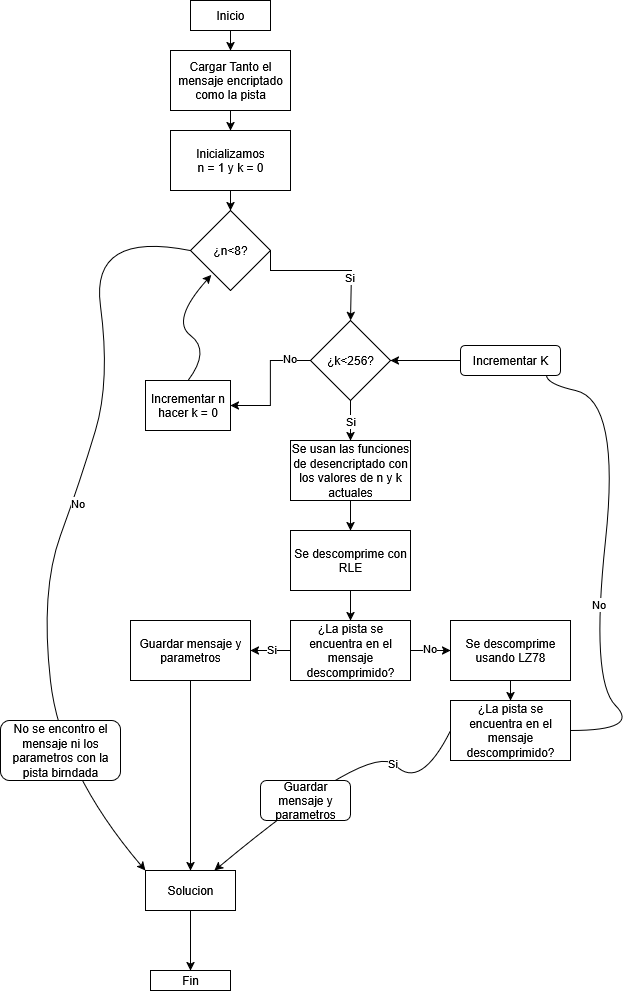
jose.rendon1@udea.edu.co

## **Análisis del Problema**

El problema central consistió en desarrollar un sistema capaz de desencriptar y descomprimir archivos binarios sin conocimiento previo de los parámetros de encriptación ni del método de compresión utilizado. El análisis reveló que todos los archivos comprimidos y encriptados seguían una estructura uniforme de ternas de 3 bytes, con dos posibles métodos de compresión (RLE y LZ78) y un sistema de encriptación doble que combinaba rotación de bits y operación XOR con una clave.

Las consideraciones técnicas incluyeron el diseño de una arquitectura modular que separara las responsabilidades en tres componentes principales: utilerías para manejo de archivos e interfaz de usuario, encriptación para la lógica criptográfica y compresión para los algoritmos de descompresión. Se establecieron restricciones como un rango de rotación de 1-7 bits para evitar operaciones sin efecto, claves de 0-255 correspondientes a 1 byte, y límites de memoria para garantizar eficiencia en el procesamiento.

**Diseño de la Solución**



## **Esquema de Tareas del Desarrollo**

El desarrollo se organizó en tres fases principales. La primera fase de análisis y diseño incluyó el análisis de requisitos y la definición de funciones. La segunda fase de implementación comprendió el desarrollo y creación del manejo de archivos e interfaz de usuario, la parte de encriptación con algoritmos de rotación y aplicación de XOR, y la instancia de compresión con descompresión RLE y LZ78. La tercera fase de integración y pruebas abarcó la integración de módulos, pruebas con el dataset de desarrollo y optimización con manejo de errores.

## **Funciones Implementadas**

Se implementaron cuatro funciones principales. El algoritmo de desencriptación utiliza rotación circular a la derecha y operación XOR, procesando cada byte mediante la secuencia: aplicación de XOR con la clave y rotación circular a la derecha. El algoritmo de descompresión RLE procesa ternas donde el primer byte se ignora como basura, el segundo representa las repeticiones y el tercero el carácter a repetir, expandiendo cada carácter según el número de repeticiones especificado.

El algoritmo de descompresión LZ78 procesa ternas donde los dos primeros bytes forman un índice de posición y el tercer byte representa un carácter nuevo, manteniendo un diccionario dinámico de patrones y construyendo la salida mediante concatenación de entradas del diccionario. El algoritmo de búsqueda por fuerza bruta prueba todas las combinaciones de rotación (1-7) y clave (0-255), aplicando desencriptación y probando ambos métodos de descompresión hasta encontrar la combinación que produce texto que contiene la pista conocida.

## **Problemas de Desarrollo Enfrentados**

El desarrollo enfrentó varios desafíos significativos. La gestión de memoria en LZ78 presentó problemas de desbordamiento de diccionario, solucionado mediante la implementación de límites seguros (65,536 entradas máximas), capacidades adaptativas basadas en el tamaño de entrada y redimensionamiento del diccionario. La interpretación de ternas en RLE generó confusión sobre el uso del primer byte, resuelto a tiempo gracias a la información que especificaba el formato de basura-repeticiones-carácter para RLE.

La eficiencia en búsqueda por fuerza bruta representaba un riesgo para la eficiencia con 1,792 combinaciones posibles por método de descompresión, optimizado mediante salida temprana al encontrar solución y validación de tamaños antes del procesamiento.

## **Evolución de la Solución y Consideraciones de Implementación**

La solución evolucionó desde un enfoque inicial erróneo, a uno que dividía el problema y permitia un análisis progresivo con la gran ventaja de un punto de partida claro (Desencriptación).

Las consideraciones clave de implementación incluyeron un patrón estricto de gestión de memoria basado en asignación-verificación-liberación, validación de entradas, capacidades adaptativas que ajustaban la memoria según el tamaño de entrada.

Las lecciones aprendidas enfatizaron la importancia de documentación temprana de formatos de datos, pruebas incrementales, liberación inmediata de memoria después del uso y validación de entradas y estados. Para mejoras futuras se identificó la prueba de combinaciones, desarrollo para detección de método de compresión, implementación de interfaz gráfica y generación de reportes detallados del proceso.

La solución final representa un sistema que resuelve el problema de desencriptación y descompresión dentro de los parámetros establecidos, demostrando la evolución desde un concepto inicial hasta una implementación sólida.