**INFORME FINAL DESAFIO-I INFORMÁTICA-II**

**Análisis del problema:**

El problema planteado consiste en un texto encriptado en un archivo .txt, la encriptación sigue un proceso donde en primer lugar se le hace una compresion que se puede realizar de dos tipos:

1. **RLE:** El método de compresión RLE agarra los caracteres de un texto como lo puede ser “aabbbcccc” y lo comprime de la forma (numero de repeticiones de caracter,caracter), dando como resultado algo de la forma 2a3b4c.
2. **ZL28:** transforma un texto en una secuencia de pares (índice, símbolo), mientras va construyendo un diccionario dinámico de subcadenas ya vistas.

Luego el mensaje comprimido se somete bajo un procesos de encriptación que sigue dos metodos que se aplican uno despues del otro, estos son:

1. **Cifrado XOR:** El cifrado XOR usa una clave que le sirve como método de comparación, entre el carácter y la clave, este método consiste en comparar bit por bit del carácter con un bit de la clave mediante la operación XOR y reemplazar usando el bit resultante, lo especial que tiene este método es que si se hace dos veces con la misma clave este devuelve el mensaje original.
2. **Rotación de bits hacia la derecha:** La rotación de bits es muy simple, está solo consiste en rotar los bits un número n de veces hacia la derecha dando así un carácter totalmente distinto al original.

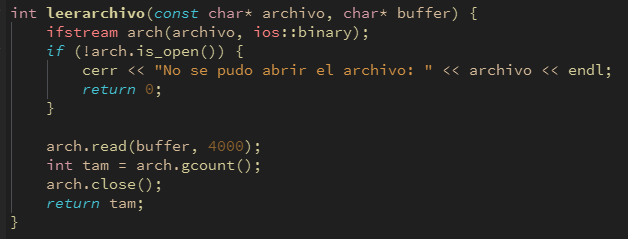
Entonces el objetivo es mediante el lenguaje c++ desarrollar un programa el cual reciba una cantidad X de archivos con un mensaje encriptado y los desencripte, esto sin saber la clave XOR, el número n que se usó en la rotación de bits y el método de descompresión usado. Esto se lleva a cabo mediante el uso de ingeniería inversa para así efectuar la inversa de los procesos llevados a cabo, aplicando asi un descifrado POR, un movimiento de bits hacia la izquierda y por último una descompresión.

Además en el desarrollo del programa se debe hacer un óptimo uso de la memoria dinámica, también se debe de cumplir con ciertos requerimientos, como lo son imprimir en consola el mensaje desencriptado y la información de los parámetros usados al momento de encriptar, osea, la clave XOR, el número n en la rotación de bits y el método de encriptación usado.

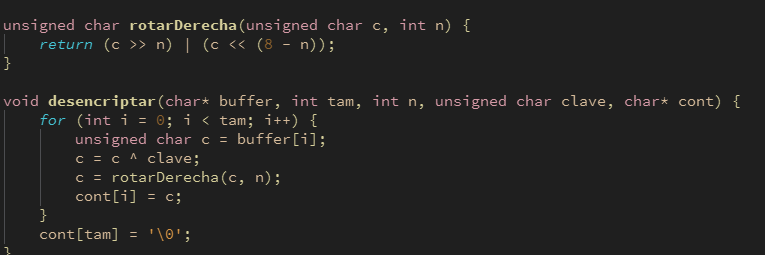
**ESQUEMA DE LA IMPLEMENTACION:**

**ALGORITMOS IMPLEMENTADOS:**

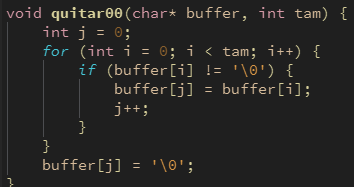
1. **Lectura de archivos:**

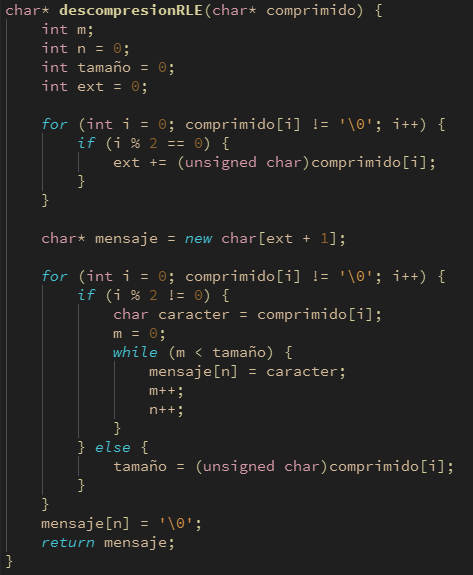
****La lectura de archivos consta de un algoritmo simple, este recibe el nombre del archivo a abrir y un arreglo donde va a poner el contenido del archivo, abre el archivo en modo binario, luego hace una validacion de que archivo haya sido abierto y copia el contenido el contenido del archivo al char de salida, ademas calcula el tamaño del archivo y lo devuelve

1. **Desencriptado:**

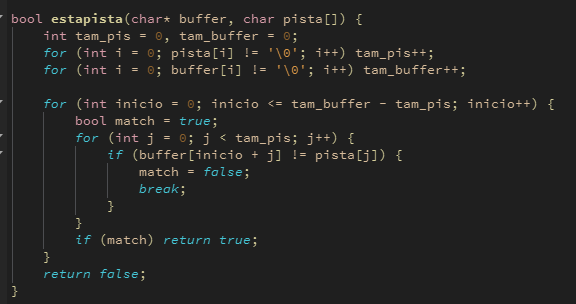
****Recibe un arreglo a desencriptar, el tamaño del arreglo, el numero de la rotación de bits, la clave xor, y el nuevo arreglo donde se va a almacenar el mensaje desencriptado, el funcionamiento de este algoritmo trata de tratar byte por byte del arreglo, primero aplicando la operación xor y luego haciendo una rotación de bits hacia la derecha, para después guardar ese bit tratado en un nuevo arreglo.

1. **Quitar bytes basura:**

****Esta funcion esta destinada a limpiar los bits basura que hay dentro del arreglo, para que asi a la hora de usar descompresion de RLE haya un correcto funcionamiento, sus parametros de entrada son: arreglo con el mensaje con basura y el tamaño del arreglo, este recorre byte por byte y solo almacena los que son distintos a basura modificando asi el arreglo original.

1. **Descompresion RLE:  
     
   **La función recibe el texto a descomprimir y devuelve el texto descomprimido, lo primero que hace es calcular el tamaño del arreglo de salida, esto sumando el valor de todos los índices, luego en un for guarda la información de cada índice como el tamaño de la secuencia, y luego el carácter lo guarda en el nuevo arreglo la cantidad de veces que indique el tamaño.
2. **Descompresion LZ78:  
   **La función parsear LZ78 recibe un flujo binario comprimido y lo divide en bloques de tres bytes: dos se usan como índice y uno como carácter. Con esta información se generan dos arreglos paralelos, uno con los índices y otro con los caracteres, que representan la secuencia comprimida en una forma más manejable.

Por su parte, descomprimir LZ78 toma esos pares (índice, carácter) y reconstruye el mensaje original utilizando un diccionario dinámico. En cada paso copia la cadena asociada al índice y le añade el nuevo carácter, tanto al resultado como al diccionario. De esta manera, se van generando progresivamente las cadenas que conforman el texto original.

1. **validación pista en desencriptado  
   **Esta funcion logra comparar caracter por caracter de la pista y de una porcion del texto desencriptado para ver si coinciden, la porcion del texto de va recorriendo por todo el texto hasta llegar a la porcion indicada, la porcion siempre es del mismo tamaño de la pista, devuelve true si la pista esta
2. **Algoritmo principal  
   **El algoritmo principal es donde se ejecuta el programa, este primeramente lo que hace es preguntar cuántos archivos se van a leer, luego entra a un while donde se va a ejecutar el programa con cada uno de los archivos, se leen los archivos y se guardan en variables para luego entrar en un for, este primer ciclo recorre key por key del XOR, luego entra a otro for el cual da una valor n a la rotación de bits, desde 0 a 7, para luego validar si alguna descompresión es correcta con algunos de estos parámetros, así logrando verificar cada una de las posibles combinaciones,si no se encuentra ninguna coincidencia con ninguna combinación el programa imprime el error de que no se ha logrado hacer la desencriptación, al momento de encontrar cual es la combinación correcta imprime los parámetros necesarios en pantalla y termina los ciclo for para así seguir con el siguiente archivo

**PROBLEMAS DE DESARROLLO AFRONTADOS:**

* El primer problema y principal que afrontamos a la hora de desarrollar el desafío fue con la plataforma git, ya que uno de los integrantes del equipo no sufrió varios inconvenientes a la hora de clonar el repositorio y hacer push, primeramente hubo problemas clonando, por lo cual se tuvo que realizar cambios en la configuración de redes de la máquina, de igual manera al intentar hacer push siempre se presentaban dificultades, por lo cual este integrante solo pudo hacer 1 commit.
* El segundo problema afrontado vino en la elaboración del desencriptado RLE, ya que en ese momento no teníamos conocimiento de que la encriptación venía en forma de ternas, por lo cual no contábamos que el primer byte era basura, después de analizar a profundidad el problema se llegó a la conclusión de que la solución más óptima que pensamos fue eliminar los bytes basura en otra función
* El tercer y último problema fue relacionado a la desencriptación con el método LZ78, debido a que en primera instancia se realizo el uso de una función que contaba con el uso de una estructura, lo cual no estaba permitido, por lo cual se tuvo que implementar otro algoritmo distinto con un nivel de complejidad mayor, pero que de igual manera funciona.

**EVOLUCIÓN DE LA SOLUCIÓN:**

Primeramente en el desarrollo de la solución planteamos un diagrama de flujo algo básico que nos sirvió como base para el desarrollo final, aunque primeramente hicimos cada una de las funciones y nos aseguramos de que funcionaran cada una, esto mediante pruebas con los parámetros necesarios para cada función, luego cuando ya tuvimos todas las funciones hechas pasamos al desarrollo de un diagrama más detallado donde ya pudimos montar de manera correcta todo el código base,  
  
Las consideraciones en el programa, si las keys del XOR usadas son números muy grandes la cantidad de operaciones realizadas por el programa se vuelve muy grande, además de que si se usa un archivo muy grande puede haber un overflow en la lectura del archivo, entonces se tendría que cambiar el tamaño del arreglo donde se leen los archivos.  
  
**DIAGRAMA DE FLUJO:**

