INFORME DE PARCIAL 1

Informática II

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

En este escenario, se requiere desarrollar un programa, el cual, debe simular el funcionamiento de una cerradura basada en una regla K dada, se deben de cumplir las condiciones establecidas en la regla; por medio de matrices y valores numéricos que van a hacer el papel de regla o clave.

REQUISITOS DEL PROGRAMA:

- M es una matriz neutra, donde el elemento del centro (entre filas y columnas) está vacío, tiene una dimensión impar (para poder tener un centro) y los elementos alrededor se llenan con números de la forma n_anterior+1, en el orden secuencial, de izquierda a derecha y viceversa, siempre apuntando hacia abajo. M se puede rotar. (CONSIDERAR SI HAY ALGÚN PROCESO PARA HALLAR LA "TRANSPUESTA" DE M Y PODERLA ROTAR MÁS FÁCIL).

- Para las cerraduras X, se usan varias M, alineadas en el punto central (sin importar el tamaño, siempre se alinean con su elemento central).

- K es una regla, que valida las posiciones, como una entrada. K tiene unos valores que se refieren a las posiciones de todas las M que conforman a X, y se evalúan los valores allí ubicados. K tiene 5 valores, así: (fila, columna, valorA y valorB, valorB y valorC, valorC y valorD), donde, en las últimas 3 posiciones, se usa un 1, y dependiendo del signo, se define si los valores deben ser mayores o menores (1 indica valor1>valor2, -1 indica valor1<valor2).

- Para abrir la cerradura, se deben rotar las M independientemente hasta que esa regla K se cumpla.

PARTES POR DESARROLLAR:

1. Desarrollar un módulo que permita crear estructuras de datos de tamaño variable, consistentes con las características descritas en la Consideraciones Iniciales. -> Función para crear matrices de tamaño variable, usando memoria dinámica.

2. Implementar funciones que permitan realizar las rotaciones a las estructuras, tal como se muestra en la Figura 1. ->Función para rotar cada matriz (ingresada como argumento) de forma independiente.

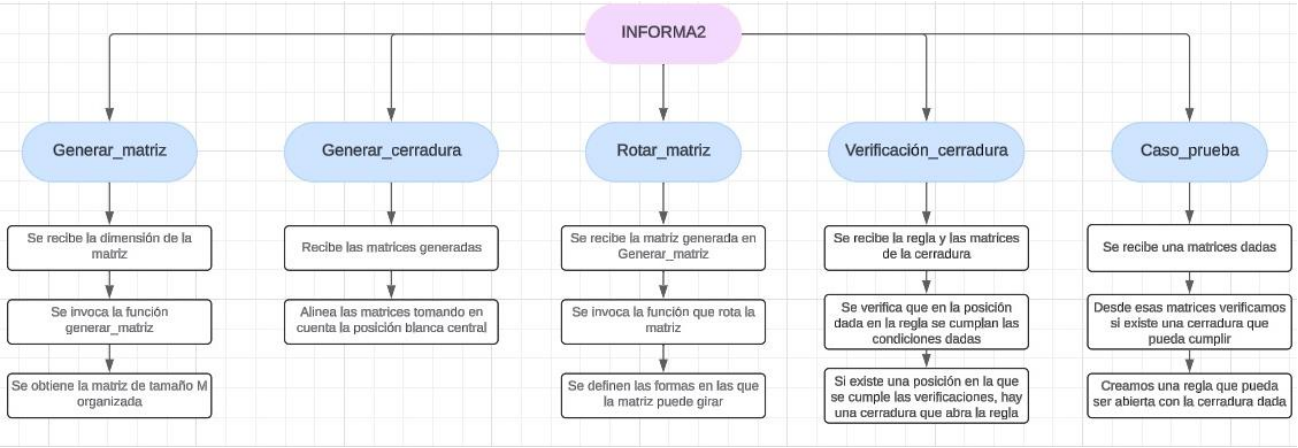
3. Desarrollar un módulo para configurar cerraduras de la tal forma que la cantidad y el tamaño de las estructuras que la componen sea variable. ->función para crear una cerradura con tamaños y cantidad de matrices variable (que el usuario pueda ingresar los valores).

4. Implementar funciones para validar una regla de apertura sobre una cerradura. -> FUNCIÓN PARA VALIDAR QUE LA REGLA INGRESADA ABRA LA CERRADURA.

5. Desarrollar un módulo para que, a partir de una regla, se genere al menos una configuración de cerradura que se pueda abrir con dicha regla. ->PROGRAMA GENERAL, JUNTO CON UNA FUNCIÓN, QUE AL EJECUTARSE, CONTENGA UNA REGLA QUE ABRA UNA CERRADURA CON VALORES YA DEFINIDOS (PROCESO Y VALORES AUTOMÁTICAMENTE ESTABLECIDOS COMO PRUEBA DEL FUNCIONAMIENTO).

b. Esquema de tareas para el desarrollo de los algoritmos.

Para la elaboración de este modelo de cerradura, se planea seguir cinco procesos generales que corresponden a las agrupaciones de tareas para las funciones del programa, así:



c. Algoritmos implementados.

Función **reservarMatriz(int m)**: Esta función se encarga de asignar memoria dinámica para una matriz de tamaño m x m. Se utiliza el operador new para asignar memoria para un arreglo de punteros que representan las filas, y luego se utiliza un bucle para asignar memoria para cada fila individualmente.

Función **liberarMatriz(int** matriz, int m)**: Esta función libera la memoria asignada dinámicamente para una matriz. Se utiliza el operador delete para liberar la memoria asignada para cada fila, seguido por delete[] para liberar la memoria del arreglo de punteros de fila.

Función **generarMatriz(int** matriz, int m)**: Esta función genera una matriz cuadrada con números ascendentes, comenzando desde 1 y colocando el valor 0 en el centro de la matriz. Utiliza dos bucles anidados para recorrer la matriz y asignar los valores correspondientes.

Función **rotarMatriz(int** matriz, int m)**: Esta función rota una matriz cuadrada en 90 grados en sentido horario. Crea una matriz temporal para almacenar la matriz rotada y luego copia los valores rotados de la matriz temporal a la matriz original.

Función **imprimirMatriz(int** matriz, int m)**: Esta función imprime los elementos de una matriz en la consola. Utiliza dos bucles anidados para recorrer la matriz y mostrar los elementos fila por fila.

Función **agregarDato(int numero, int* arreglo, int longitud)**: Esta función agrega un dato al final de un arreglo dinámico. Crea un nuevo arreglo con longitud aumentada en uno, copia los elementos del arreglo original al nuevo arreglo y luego agrega el nuevo dato al final.

Función **crearCerradura()**: Esta función principal del programa solicita al usuario una clave que define las condiciones de la cerradura. Luego genera dos matrices, una original y una para rotar. Realiza comparaciones entre los valores en posiciones especificadas por la clave en las matrices rotadas, determinando si se cumple la condición de la cerradura.

d. Problemas de desarrollo que afrontó.

Manejo de memoria dinámica: Garantizar la correcta reserva y liberación de memoria para evitar fugas de memoria o errores de acceso.

Implementación de la lógica: Desarrollar un algoritmo eficiente para crear las matrices, dependiendo de la posición indicada, agruparlas en una cerradura y rotarlas por 90 grados en sentido horario cada vez que no se cumplan las comparaciones.Validación de las entradas del usuario: Verificar que las condiciones ingresadas por el usuario sean válidas y cumplan con los requisitos del problema.

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

La solución evolucionó desde un enfoque inicial de diseño algorítmico hasta la implementación de un programa funcional que cumple con los requisitos del problema. Algunas consideraciones importantes para la implementación incluyen:

- Garantizar la eficiencia en el uso de la memoria y el tiempo de ejecución.
- Validar las entradas del usuario para evitar errores de ejecución.
- Documentar adecuadamente el código para facilitar su comprensión y mantenimiento.
- Realizar pruebas exhaustivas para verificar la corrección y robustez del programa en diferentes escenarios.
- La necesidad de tratar cada matriz M de la cerradura X como un elemento independiente, con la posibilidad de hacer rotaciones de forma arbitraria sobre cada M, sin necesidad de influir en el estado actual de sus semejantes.

Con estas consideraciones en mente, se logró desarrollar una versión del programa que genera la configuración de cerradura requerida y proporciona información sobre las rotaciones necesarias para abrirla. Posteriormente, es posible mejorar y completar el programa, adicionando el aumento de dimensiones para las matrices y la salida final donde se indique las dimensiones y modos (o rotaciones) de una cerradura que se abra con la clave que ingresa el usuario.