**Logotipo

Descripción generada automáticamente**

**Desafío 1**

Estudiante

Luis miguel Espitaleta Alvarez

1047471147

Tutor  
Augusto Enrique Salazar Jimenez

Universidad de Antioquia  
Facultad de Ingeniería

Medellín, Antioquia, Colombia

**Informa del Proyecto (Desafío 1):**

**Análisis, Diseño y Implementación:** El desafío consiste en crear un sistema de cerradura y llave utilizando matrices cuadradas. La cerradura está compuesta con un conjunto de matrices cuadradas de diferentes tamaños, cada una con llenas con números exceptuando el centro, mientras que la llave es una secuencia de valores que se utilizan para comparar y determinar si la cerradura puede ser "abierta" o no la cual esta compuestas por un arreglo de enteros, donde los primeros dos números nos señalan la posición a comparar entre dos matrices de la cerradura y los séquiatenos números que pueden ser 0, 1 o -1 nos indican la regla de la cerradura siendo el caso donde M1 > M2 entonces 1 seria valido y si M1 < M2 entonces -1 seria valido y en el caso que M1 = M2 0 seria valido.

Ya teniendo en cuenta es te breve análisis del problema, el primer diseño para la solución de este desafío seria dividir el problema en subproblemas más manejables, como la creación de las matrices para la creación de la cerradura, la comparación de matrices la cual se dividiría aún más porque tendría que solucionar el problema de rotar la matriz y la interacción con el usuario.

Con todas estas consideraciones, implementaría funciones como crearMatriz(), la cual me permitirá crear una matriz cuadra pidiendo el tamaño de la matriz pero como la matriz es cuadrada solo necesitaríamos un parámetro, otra función que implantaría seria compararMatrices() la cual tendría dentro un llamado de otra funciona que sería rotarMatriz() sería lo primero que me llega a la cabeza con el análisis.

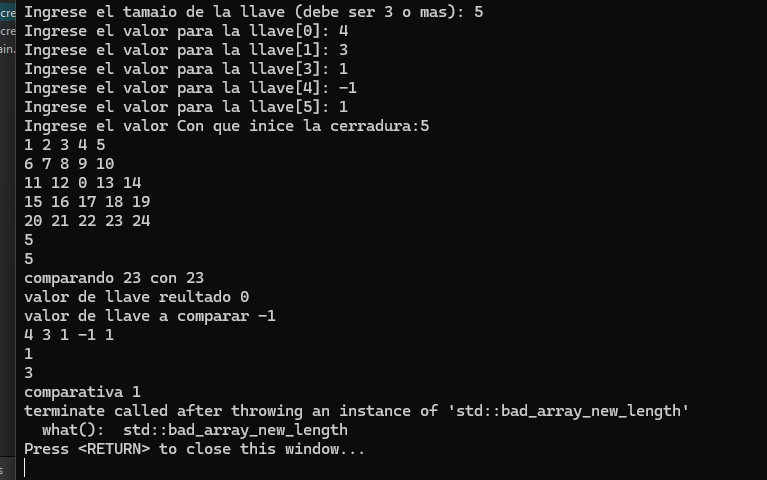
**Problemas en el desarrollo:** El primero problema que tuve fue hacer código sin seguir los pasos del desafío ya tengo un código hecho el cual debe estar en desorden ya que tengo las ideas un poco mas ordenadas luego de implementar el análisis.

El momento de hacer el Código para crear la cerradura, presente un atasco al buscando la manera de rellenarlo validando los valores de la llave, ya que le pedí el primer valor de la cerradura al usuario y que las demás las llenara validando la llave, cree un for que reccora la llave pregunte al usuario por le primera nuero del arreglo y luego vaya haciendo la comparación de matices peor al momento de imprimir quedo atascado Texto

Descripción generada automáticamente

Debería decir comparando N de la M1 con N de la M2 habia un error al rellenar la matirz pero era poque estaba iterando dos veces la variable i, el la función comparar matrices al momentor de hace el ciclo tenia el mismo error iterando dos veces i

Ahora el problema que se me acaba de presentar es cuando hacer comparación del resultado de comparación matrices con el vale 3 de la llave



El error se estaba presentando al rotar la matriz porque no estaba mandando correctamente el tamaño de la matriz.

Cuando pase el código de llenado manual del primer campo a manera automática empecé a presentar problemas cíclicos

Imagen que contiene Texto

Descripción generada automáticamente

Había retirado la variable para pedir el valor inicial del array pero no lo había quitado de a la validación del do While

Ya haciendo validaciones con matrices arriba de 5 me genera problemas sigue un ciclo infinito, luego de resolver este problema me doy cuenta de que la mayoría de errores que he tenido cíclicos es por malas comparaciones

Al ogranizar ya el código para dejarlo documentado cambie la la variable de tamaño cerradura ya que estaba pasando el parámetro como (tamanioLlave-1) y lo que hice fue asignarle ese valor a tamanioCerradura y emepzo a imprimir las matrices de manera cíclica, bueno el error estaba en la inicialización de la variable (tamanioCerradura) como la estaba definiendo (tamanioLlave-1) tenia que ser después de pedir la variable tamanioLlave

**Evolución de la solución: T**engo un archivo donde está todo el Código donde el Main tienen código base que me pregunta el tamaño de la cerradura, si quiero rellenar el Código manual o automático luego se reserva en memoria el arreglo de matrices para después en un bucle **For** se rellena cada matriz del arreglo cerradura para después en otro bucle **For** empezar hacer las comparaciones todo este Código quedara en el historial del git.

A partir de aquí registrare los problemas que enfrente al organizar el Código que ya tenía hecho y hacer el faltante, lo primero es organizar las funciones en un módulo, teniendo en cuenta el primer punto que es “Desarrollar un módulo que permita crear estructuras de datos de tamaño variable, consistentes con las características descritas en la Consideraciones Iniciales” entiendo que es pedirle a usuario que cree la llave y se procede a crear el modulo de crear llave tomando Código que ya tenía en el main.

Luego segui creando el modulo de crear cerradura con el cual presente muchas más dificulta porque presente vario errores aun colocando el mismo código que ya tenía previamente, aunque llenaba la cerradura correctamente validando la llave de ejemplo no daba el mismo resultado esperado que era (5, 7, 5, 9)

Texto

Descripción generada automáticamente

Adicionando un if adicional para validad que a partir del 3 digito también en esa posición valide dos posiciones antes y no coloque el mismo número de matriz

El primer campo lo estaba llenado de manera manual al querer pasarlo a manera automática empecé a presentar problemas de un ciclo interminable, estaba validando una variable que ya había retirado

Ya luego de limpiar el código de validaciones procederé a limpiar la memoria.

**Algoritmos implementados:**

**main():** Se declaran las variables para el tamaño de la llave (tamanioLlave), tamaño de la cerradura (tamanioCerradura) el cual se le asigna restándole uno a la variable (tamanioLlave), se utilizan punteros para mantener referencias a estas variables (ptrTamanioLlave y ptrTamanioCerradura). Solicita al usuario que ingrese el tamaño de la llave, asegurándose de que sea al menos 3 mediante un bucle do-while. Crea dos arrays dinámicos (llaveK y cerraduraK) basados en el tamaño de la llave y la cerradura.

Luego se llaman las funciones llenaLLave() y llenaCerradura() que estan definidas en "modulos/crearCerradura.h" y "modulos/crearReglaK.h" respectivamente para llenar los arrays llaveK y cerraduraK.

Finalmente se imprimen los valores de llaveK y cerraduraK cada uno mediante un for que itera en cada posición del arreglo en la consola y ya como último se libera la memoria asignada dinámicamente a llaveK y cerraduraK utilizando delete[].

**llenaLLave():**La función llenaLLave se encarga de llenar un array con valores de una clave (llaveK) ingresados por el usuario. Antes de asignar cada valor al array, realiza ciertas validaciones para asegurarse de que los valores ingresados estén dentro de los límites establecidos.

**imprimirMatriz():** Esta función por medio de dos for anidados imprime una matriz cuadrada en la consola. Recibiendo los parámetros **matriz**: Matriz cuadrada de enteros que se va a imprimir. **tamano**: Tamaño de la matriz (número de filas y columnas) se pasa un solo valor por que la matriz es cuadrada.

**llenaCerradura():** Esta función llena un arreglo (cerraduraX) con valores basados en operaciones con matrices y comparaciones que viene de llamar **compararPosicionMatrices()**. Recibe los parámetros: tamanioCerradura, llaveK. Luego retorna un puntero a un arreglo de enteros (cerraduraX) que ha sido llenado según las operaciones realizadas.

**crearMatrizCuadrada():** Esta función crea una matriz cuadrada con valores numéricos. Recibiendo parámetros **tamano:** Tamaño de la matriz (número de filas y columnas) y regresa un puntero a una matriz cuadrada recién creada.

**rotarMatrizCuadrada():** Esta función rota una matriz cuadrada en sentido horario (90 grados). recibe parámetros **matriz:** Matriz cuadrada que se va a rotar.

**tamano**: Tamaño de la matriz y retorna un puntero a una nueva matriz cuadrada que es la matriz original rotada 90 grados.

**compararPosicionMatrices():** Esta función compara una posición específica (fila, columna) de dos matrices (matrizA y matrizB).

Recibe los parámetros:

**matrizA:** Primera matriz cuadrada.

matrizB: Segunda matriz cuadrada.

tamanoA: Tamaño de la matriz matrizA.

tamanoB: Tamaño de la matriz matrizB.

llave: Arreglo de enteros que contiene información clave para la comparación.

Retorna:

1 si el elemento en la posición de matrizA es mayor que el de matrizB.

-1 si el elemento en la posición de matrizA es menor que el de matrizB.

0 si los elementos en la posición son iguales.

2 si ocurre algún error en los límites de las matrices.

**liberarMemoria():** Esta libera la memoria asignada dinámicamente para un arreglo de matrices.