INFORME EXAMEN PARCIAL 1 – INFORMATICA 2 – 2024-1

PARTICIPANTES:

* Jhon Stiven Herrera Herrera
* Miguel José Vargas Pacheco

FECHAS DE ENTREGAS:

* 29 de marzo: proceso de análisis y diseño de la solución.
* 5 de abril: proceso de implementación

REQUISITOS:

* Genere un informe en donde se detalle el desarrollo del proyecto (instrucciones en el desafío)
* La solución debe ser implementada en lenguaje C++.
* La implementación debe incluir el uso de punteros, arreglos y memoria dinámica.
* Se debe crear un repositorio público para cargar todos los archivos relacionados con la solución (informe, código fuente y otros anexos).
* Una vez cumplida la fecha de entrega no se podrá hacer modificación alguna al repositorio.
* Se deben hacer commits de forma regular (al menos dos al día) de tal forma que se evidencie la evolución de la propuesta de solución y su implementación.
* Se debe adjuntar un enlace de YouTube a un video con las instrucciones planteadas en el desafío.
* Cumplir con el plazo de entrega establecido.
* Se deben adjuntar dos enlaces: uno al repositorio y otro al video, nada más.
* Para la evaluación del desafío se realizará una sustentación oral en un horario concertado con el profesor. La asistencia a la sustentación es obligatoria.

RESTRICCIONES:

* No se debe utilizar librerías.

ESPECIFICACIONES:

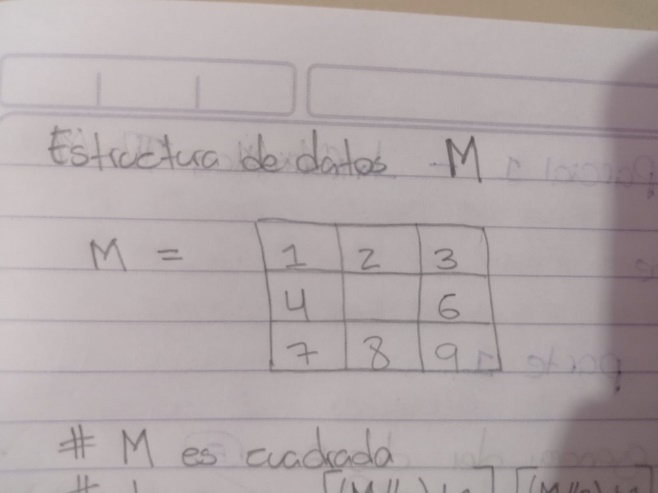
* La empresa Informa2 tiene la necesidad de que dada una regla K, se genere una configuración de X que la satisfaga. La salida del programa debe ser la configuración X, tamaño de cada dimensión y rotaciones que se deben hacer en cada estructura para abrir la correspondiente cerradura.

EVALUACION:

* [10%] Desarrollar un módulo que permita crear estructuras de datos de tamaño variable, consistentes con las características descritas en la Consideraciones Iniciales
* [10%] Implementar funciones que permitan realizar las rotaciones a las estructuras
* [10%] Desarrollar un módulo para configurar cerraduras de la tal forma que la cantidad y el tamaño de las estructuras que la componen sea variable.
* [10%] Implementar funciones para validar una regla de apertura sobre una cerradura.
* [60%] 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.

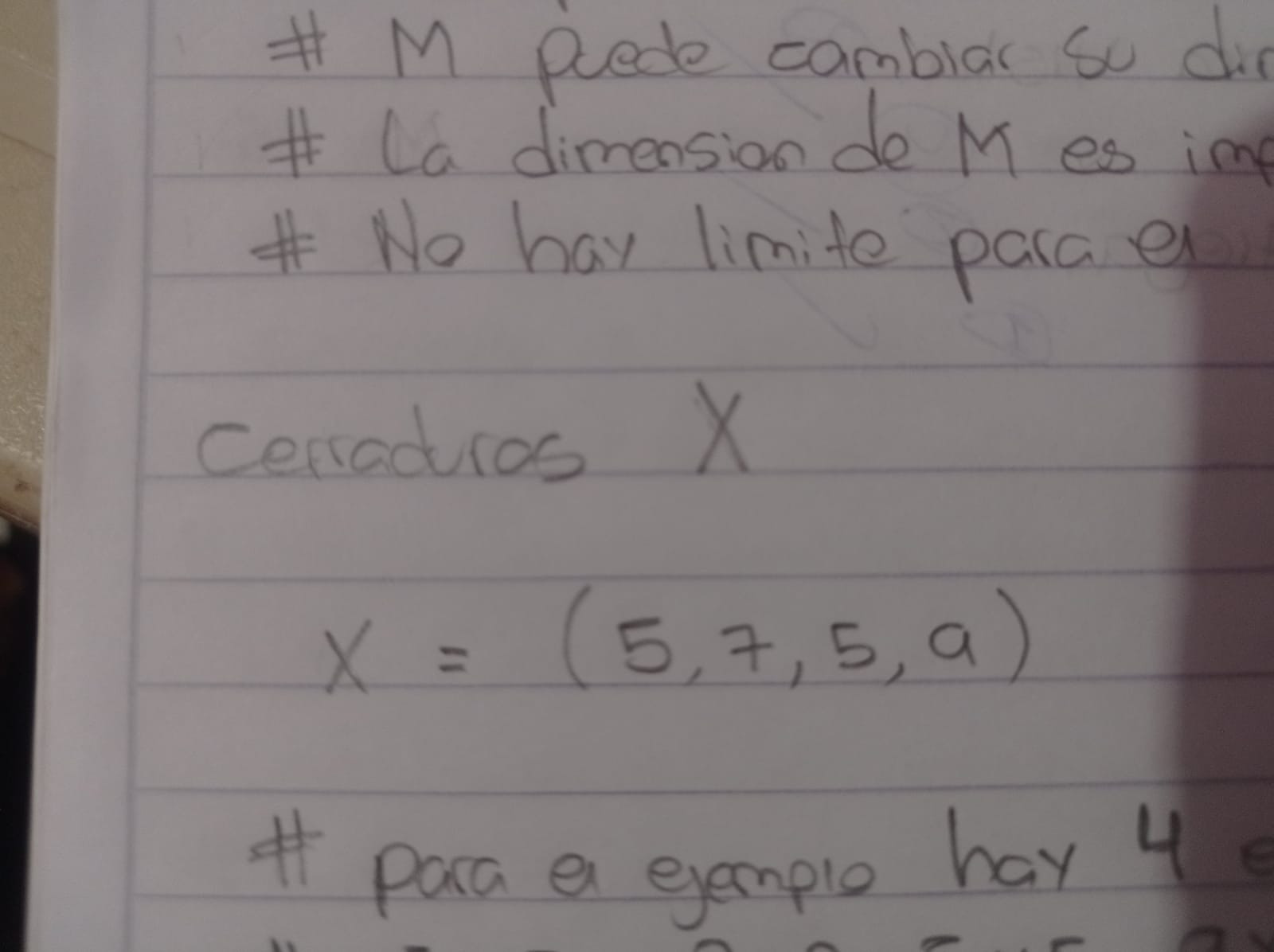
PRIMER ANALISIS DEL DESAFIO:

ESTRUCTURA DE DATOS (M)



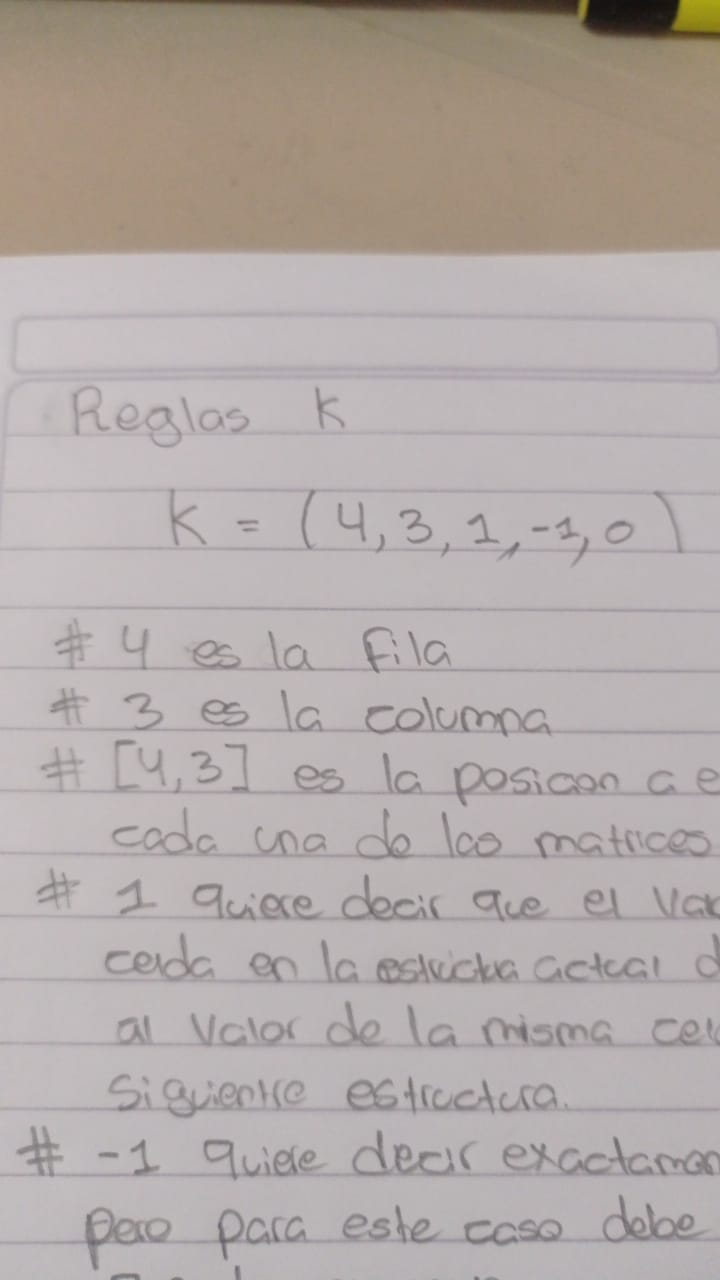
* M es una matriz cuadrada
* M puede rotar
* M rota en sentido antihorario
* M puede cambiar su dimensión
* No existe un límite para el tamaño de M
* La dimensión de M siempre es impar
* La posición del centro de M está vacía [(m/2) +1] [(m/2) +1]

CERRADURAS (X)



* X están alineadas por la celda vacía
* No hay restricciones en la cantidad de X
* No hay restricciones en el tamaño de X

REGLAS (K)



* 4 es la fila
* 3 es la columna
* [4,3] es la celda para evaluar en cada una de las matrices
* 1 quiere decir que el valor de la celda en la estructura actual debe ser **mayor** al valor de la misma celda en la siguiente estructura
* -1 quiere decir que el valor de la celda en la estructura actual debe ser **menor** al valor de la misma celda en la siguiente estructura
* 0 quiere decir que el valor de la celda en la estructura actual debe ser **igual** al valor de la misma celda en la siguiente estructura
* K es un booleano que para ser verdadero tenemos que rotar las estructuras con el fin de hacer que se cumpla la condición como la especifica la declaración de k

TIPOS DE DATOS A UTILIZAR:

* **MATRICES** para las estructuras de datos (M)
* **CADENA DE ENTEROS POSTIVOS** para las cerraduras (X)
* **CADENAS DE ENTEROS REALES** para las reglas (K)

REESCRIPCION DEL OBJETIVO:

* Dada una regla K genera una configuración en la cerradura X que haga que k sea verdadera, el programa debe entregar al menos una configuración que cumpla con lo anterior, debe entregar el tamaño de cada estructura y entregar la cantidad de rotaciones en cada estructura para generar esta configuración que abre la cerradura.
* Datos de entrada: regla k
* Datos de salida: configuración para abrir, tamaño de las estructuras y cantidad de rotaciones a las estructuras.

PROBLEMAS:

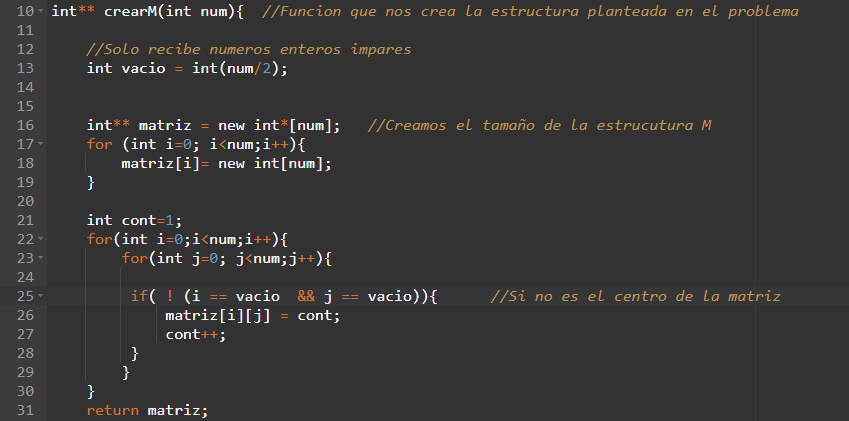
* ¿cómo usar la memoria dinámica para los cambios de dimensión de la matriz?
* ¿cómo hacer rotar la matriz?
* ¿cómo mantener vacía la posición central de la matriz? Teniendo en cuenta los cambios de dimensiones.
* ¿qué validaciones debemos tener en cuenta para el correcto funcionamiento del programa?
* ¿cómo modificar los tamaños de los tipos de datos que vamos a utilizar?
* ¿cómo saber según la regla cuantas estructuras necesito?
* Preguntar sobre el uso de librerías únicamente para el embellecimiento del programa

LISTADO DE FUNCIONES:

* Crear matriz
* Mostrar matriz
* Rotar matriz
* Longitud
* Liberar matriz
* Agregar datos
* Pedir clave
* Caso1
* Caso2

PRUEBA DE PRIMERAS FUNCIONES DISEÑADAS:

* Crear matriz (M)



Aquí tenemos el código que nos genera una matriz con un tamaño dado y siguiendo la estructura de las matrices que teníamos en el ejemplo (“desafío 1”) que serán las misma que vamos a emplear para la solución de nuestro problema, esto, haciendo uso de la memoria dinámica para el cambio de longitud y demás que se va a presentar durante la ejecución.

* Mostrar matriz

Texto

Descripción generada automáticamente

Después del correcto diseño de las matrices necesitamos, para control del código y para mostrarle además al usuario lo que está pasando, una función que nos muestre la matriz que diseñamos y las variaciones que esta va teniendo durante la ejecución del programa.

* Rotar Matriz

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Para poder encontrar una matriz que nos convenga para el desarrollo de la cerradura tendremos que rotarlas, la cerradura solo puede estar en uno de los cuatro modos (contando su estado normal) entonces esta función nos permitirá establecer la función en un modo determinado de tal forma que nos sea sencillo trabajar con esta

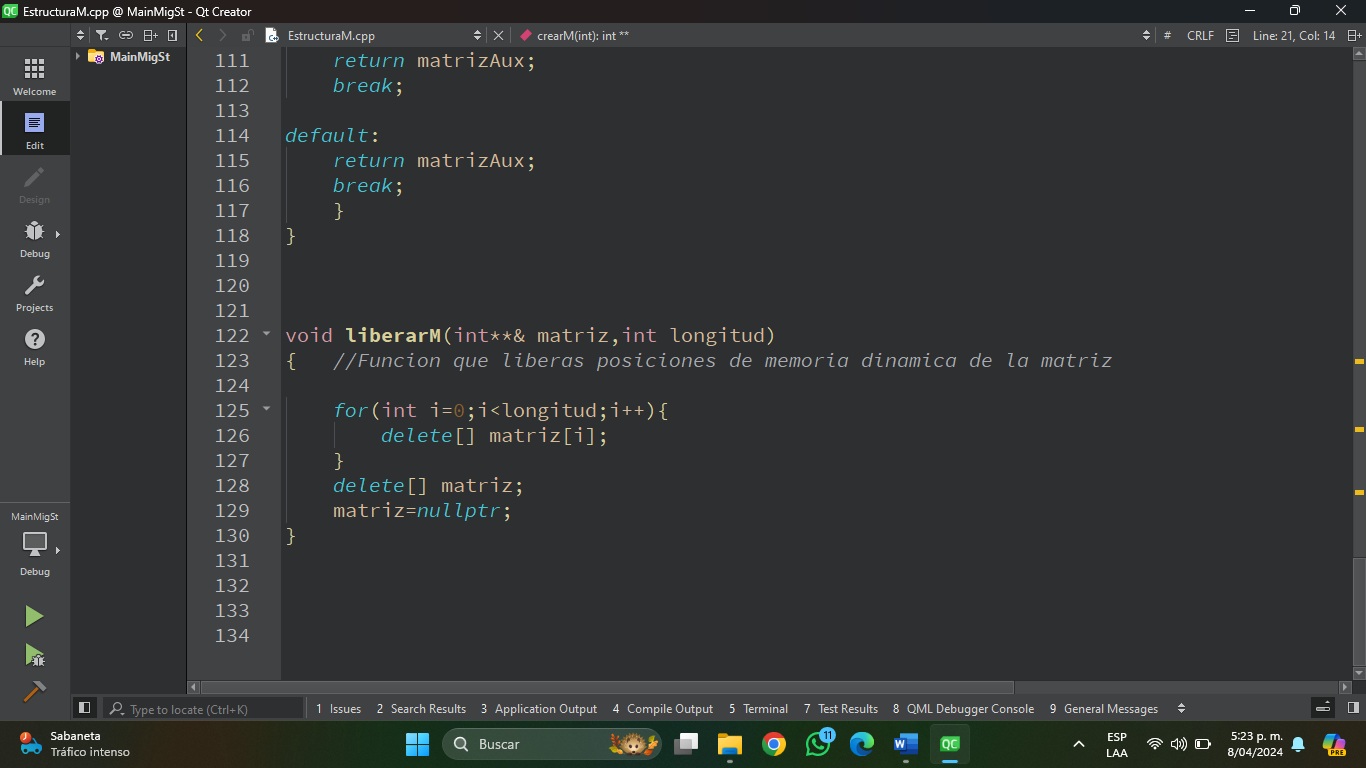
Texto

Descripción generada automáticamente

* Función de longitud

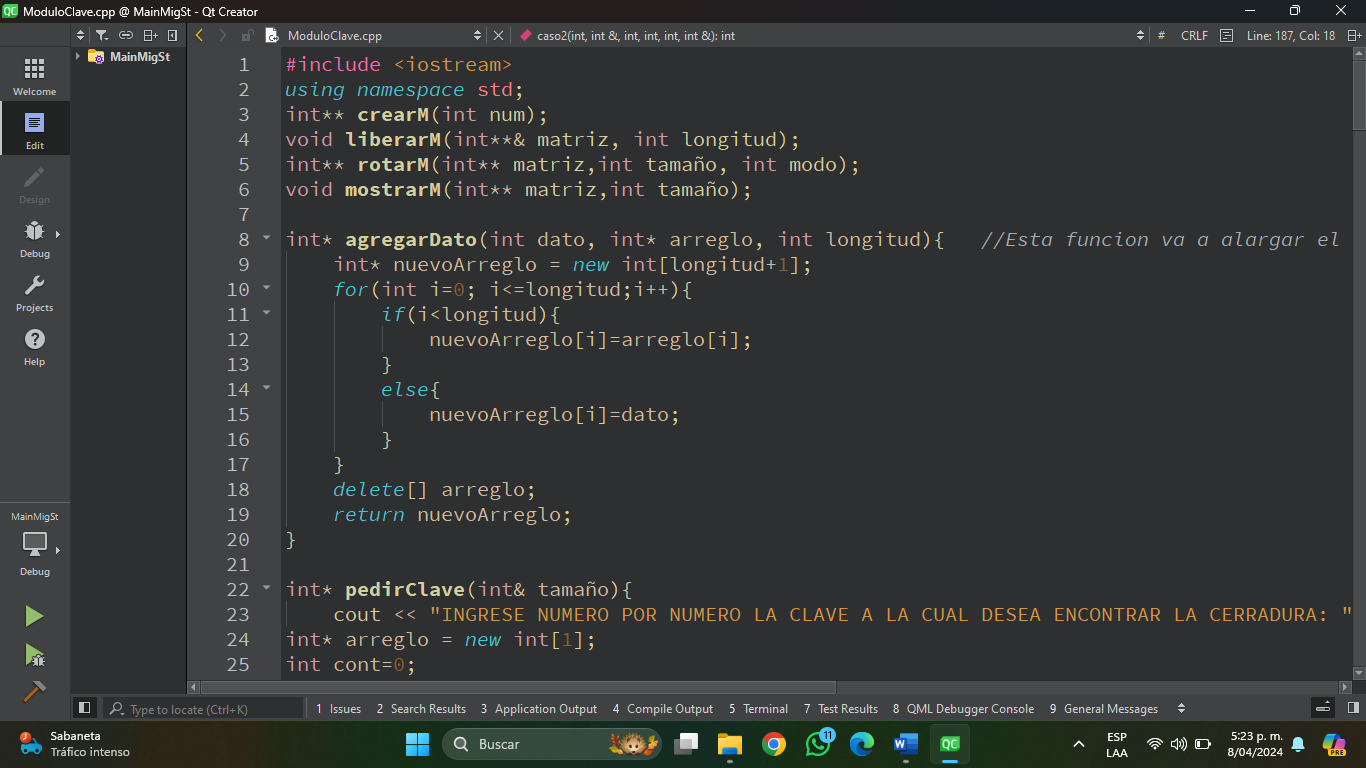
En inicio decidimos usar esta función para determinar la longitud de una dimensión de la matriz, pero luego esto nos trajo ciertos errores al implementarlo en Qt ya que nos arrojaba basura de la memoria, dañando por completo el programa, tuvimos que quitar esta implementación de todas las funciones (la mayoría) en las cuales lo implementamos.

* Liberar matriz



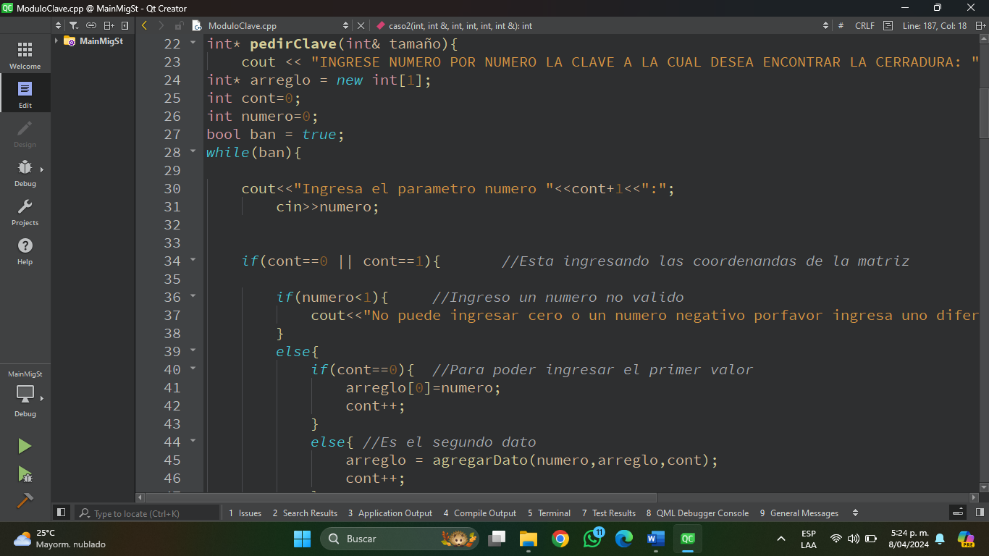
Esta funcion nos ayuda a liberar la memoria en caso de que sea necesario, de la misma manera nos permite hacer cambios en la matriz sin necesidad de guardar las variaciones en una nueva variable, de esta manera, evitamos el desperdicio de memoria.

* Agregar datos



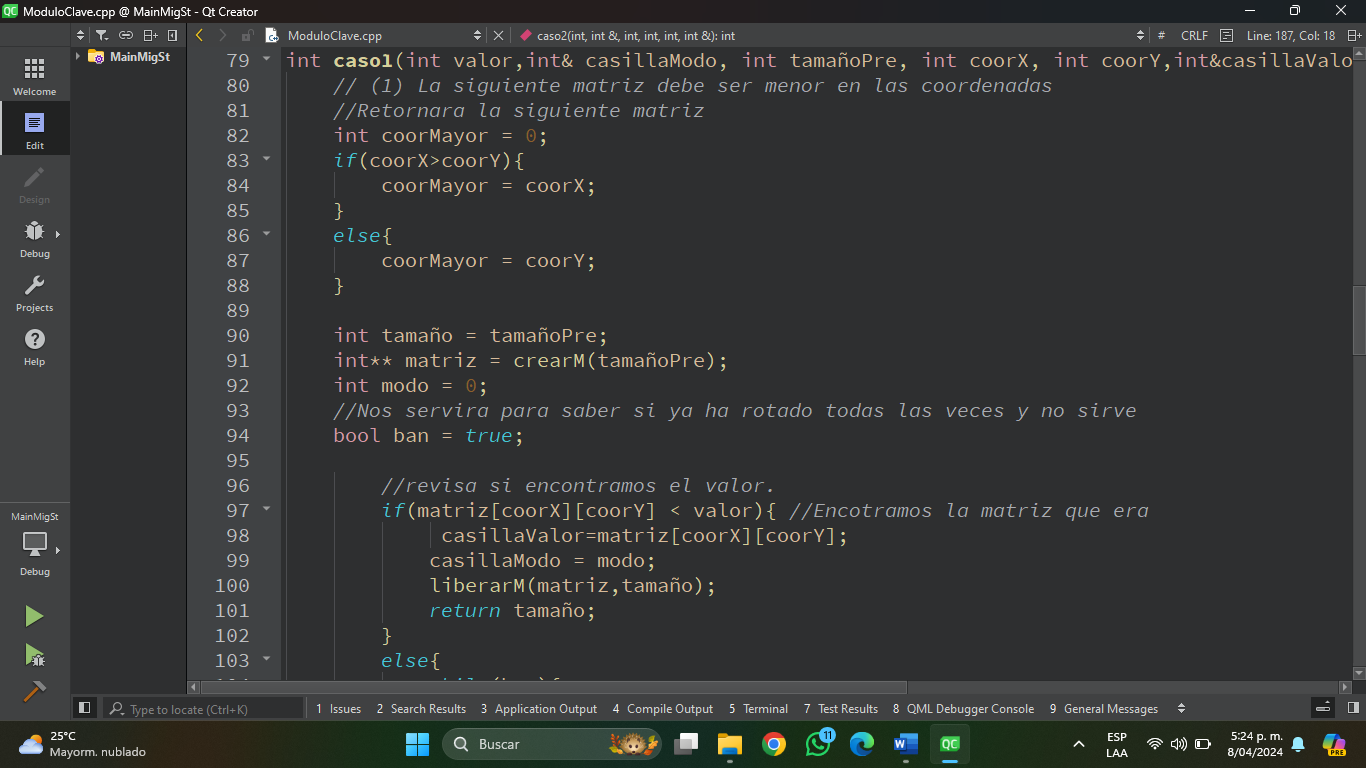
Esta es una funcion complementaria que hace parte de las funciones que nos ayudan a recibir la clave que da inicio al programa.

* PedirClave



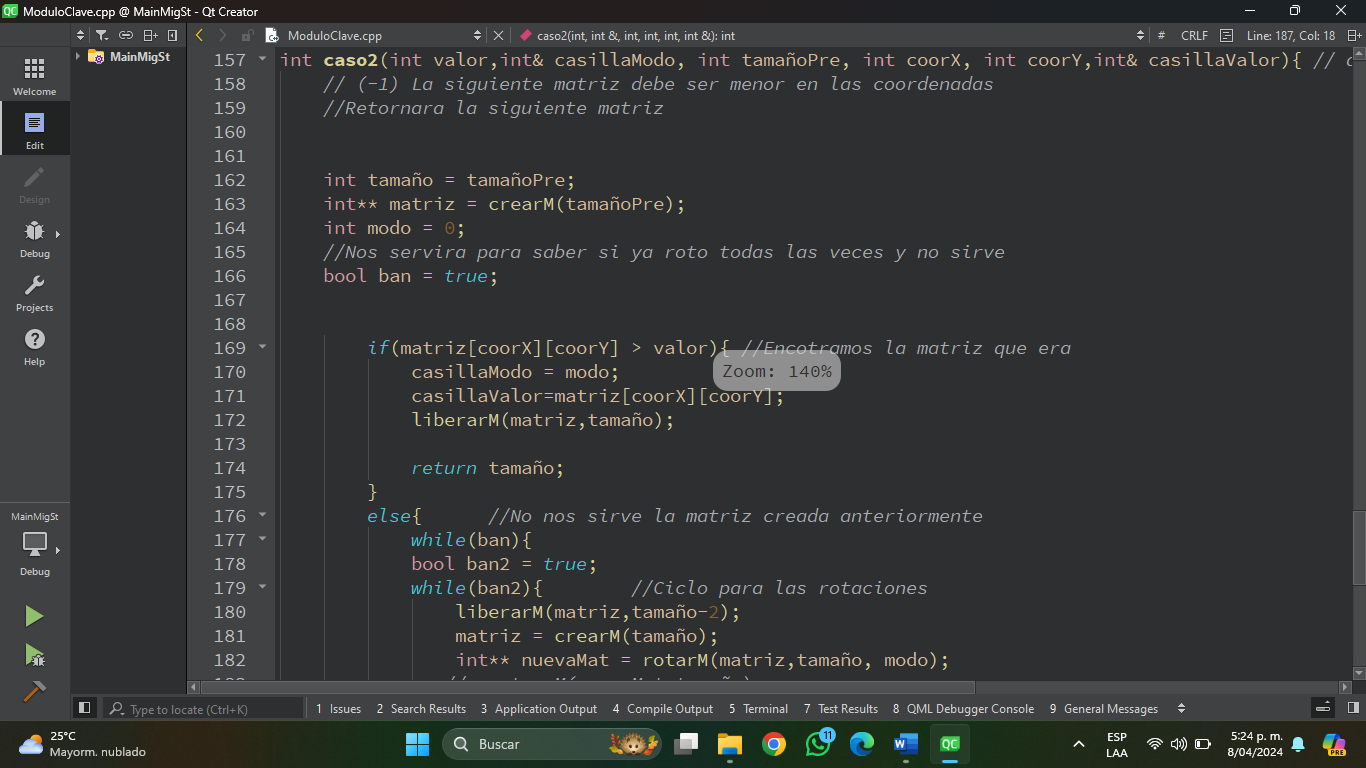
Por otra parte nuestra funcion “pedirClave” es el complemento de la anterior funcion y nos ayudan a recibir del usuario la clave.

* Caso1



Caso1 es una de las funciones del grupo de condiciones que recibimos del usuario en la clave, esta nos ayuda a crear una nueva matriz con la condicion de que en la misma celda debemos obtener un valor mayor que el guardado en la matriz anterior.

* Caso2



Caso2 es una de las funciones del grupo de condiciones que recibimos del usuario en la clave, esta nos ayuda a crear una nueva matriz con la condicion de que en la misma celda debemos obtener un valor menor que el guardado en la matriz anterior.

PERIODO DE PRUEBA:

Terminadas nuestras funciones y enlazadas en el main no queda mas que iniciar las pruebas del programa.

MOSTRA MATRIZ ---> exitosa!!

Captura de pantalla de computadora

Descripción generada automáticamente

CLAVE ---> exitosa!!

Captura de pantalla de computadora

Descripción generada automáticamente

Errores probocados a proposito en la funcion mostrar matriz

Captura de pantalla de computadora

Descripción generada automáticamente

CASO2 ---> erronea!!

la funcion caso2 nos genero una fuga en la memoria y por tanto recurrimos a rediseñarla.

