# Análisis, Desarrollo y Evolución de Algoritmos en Informática 2.

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

Para abordar el problema, dedicamos tiempo a una lectura exhaustiva del archivo desafío 1. Nuestro objetivo fue comprender a fondo el problema antes de proceder con la elaboración de cualquier diagrama de flujo o estrategia de solución. Enfocamos nuestra atención en comprender las complejidades del problema y evaluar las posibles alternativas de solución de manera integral.

Una de las principales estrategias que adoptamos fue dividir todo el programa en funciones. Esto nos permitió realizar un trabajo colaborativo más efectivo y organizado. Además, nos enfocamos en comprender cómo utilizar Git y GitHub de manera adecuada antes de comenzar con la implementación de las funciones.

Hasta el momento, hemos creado una función para cada tarea específica. Por ejemplo, una función para crear la matriz, otra para rotar la matriz y otra para determinar la posición que se está buscando en la matriz rotada, basándonos en su fila y columna. Estamos avanzando a un ritmo sólido y hemos logrado comprender muy bien el problema en su totalidad.

## b. Esquema donde describa las tareas que usted definió en el desarrollo de los algoritmos.

Las tares que definimos en cada algoritmo ósea en cada función fue recordar muy bien lo que se vio en clase acerca de arreglos, memoria dinámica, punteros y demás temas que son indispensables para el desarrollo de esta práctica, además complementándolo muy bien con lo que sea ha estudiado por nuestra propia cuenta

1. Definir la función matrizneutra ();

* Me recibe como parámetro un entero el cual va a ser el tamaño de la matriz
* De pendiendo del valor que se reciba como parámetro se llena la matriz teniendo en cuanta todas las condiciones que se nos han propuesto
* Después de usar la función se debe hacer la debida liberación de la memoria
* La función nos retorna un doble puntero

1. Definir funciones rotarmatriz1, rotarmatriz2, rotarmatriz3 ();

* Cada una recibe un entero como parámetro para determinar el tamaño de la matriz a rotar.
* Implementan algoritmos para rotar la matriz en 90, 180 y 270 grados en sentido antihorario respectivamente
* Cada una me retorna un doble puntero

1. Definir función buscar\_posicion ();

* Esta función me recibe como parámetro un puntero que tiene la dirección de memoria del primer valor de la clave la cual esta en un arreglo dinámico y el otro parámetro es un entero el cual nos determina si la siguiente matriz es mayor igual o de menor tamaño.
* Haciendo las debidas operaciones para que también apunte a la segunda dirección de memoria la cual tiene almacenada el valor de la columna
* Y después se pone en marcha un algoritmo el cual nos determina cual es el valor de la fila y columna de la siguiente matriz
* Por último, nos retorna un puntero al arreglo dinámico (cabe recalcar que se debe liberar la memoria una vez utilizada)

1. Definir saber\_dimencion\_matriz ();

* Si el valor de fila es mayor que el de columna, la dimensión de la matriz será igual al valor de fila si fila es impar. Si fila es par, la dimensión de la matriz será fila + 1
* Si el valor de columna es mayor que el de fila, la dimensión de la matriz será igual al valor de columna si columna es impar. Si columna es par, la dimensión de la matriz será columna + 1.

1. Definir comparar\_matrices ();

* Obtiene las coordenadas de la posición a comparar de cada matriz a través de los punteros clave y clave2, donde clave[0] y clave[1] representan la fila y columna de la primera matriz (ptr1), y clave2[0] y clave2[1] representan la fila y columna de la segunda matriz (ptr2).
* Compara los elementos en las posiciones especificadas de ambas matrices. Si el elemento en la posición de la primera matriz es igual al elemento en la posición de la segunda matriz, la función devuelve true, lo que indica que las matrices son iguales en esas posiciones. De lo contrario, devuelve false

1. imprimir\_matriz();

* Utiliza dos bucles for para recorrer cada fila y columna de la matriz.
* Para cada elemento en la matriz, utiliza matriz[i][j] para acceder al valor en la fila i y columna j.
* Imprime el valor de cada elemento de la matriz seguido de un tabulador (\t), lo que crea un formato de salida tabular.
* Después de imprimir todos los elementos de una fila, imprime un salto de línea (\n) para pasar a la siguiente fila.

1. Definir liberar\_memoria();

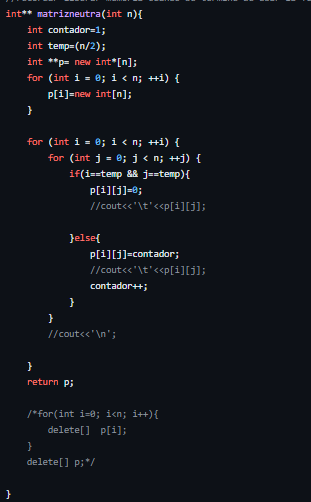
* Utiliza un bucle for para recorrer cada fila de la matriz.
* Dentro del bucle, utiliza delete ptr[i] para liberar la memoria asignada para cada fila de la matriz.
* Después de liberar la memoria de todas las filas, utiliza delete[] ptr para liberar la memoria del puntero que apunta a las filas de la matriz.

1. Por ultimo se la hace su debida implementación a cada función en el main con sus debidas verificaciones y condiciones

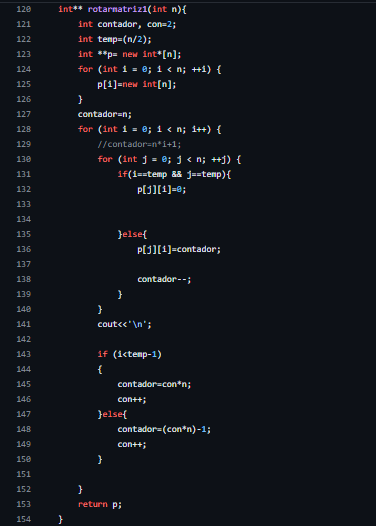
El informe se ira actualizando conforme se vaya avanzando en el main.

c. Algoritmos implementados.

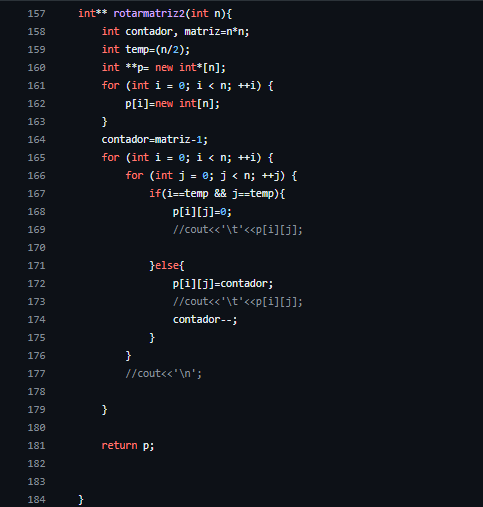
1. La función matrizneutra crea una matriz cuadrada de tamaño n e imprime sus elementos en forma de matriz, con un patrón especial que coloca el valor 0 en el centro de la matriz y luego llena los demás elementos de la matriz con números consecutivos empezando desde 1 y aumentando en orden de izquierda a derecha y de arriba hacia abajo.



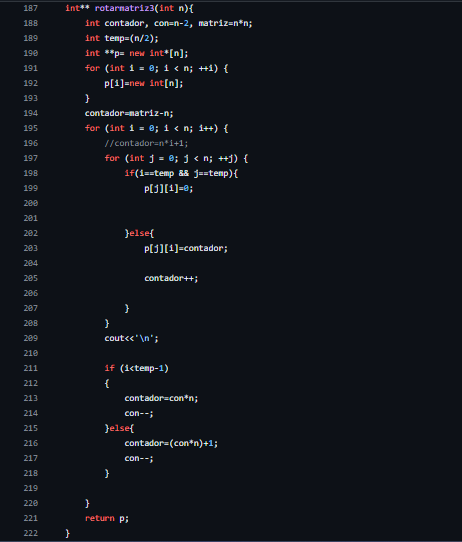
2. Esta función realiza una rotación de 90 grados en sentido antihorario de una matriz cuadrada y luego la imprime en pantalla. La lógica de rotación implica asignar valores especiales a ciertos elementos de la matriz para lograr la rotación adecuada.



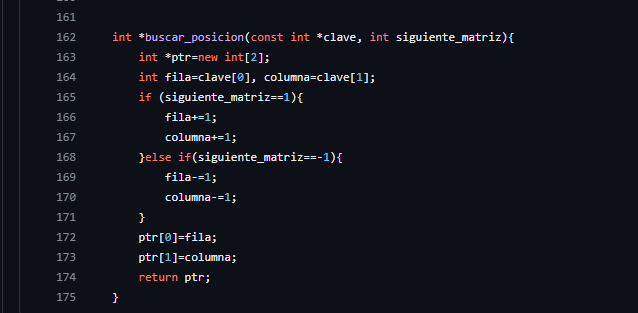
3. Esta función realiza una rotación de 180 grados en sentido antihorario de una matriz cuadrada y luego la imprime en pantalla. La lógica de rotación implica asignar valores especiales a ciertos elementos de la matriz para lograr la rotación adecuada.



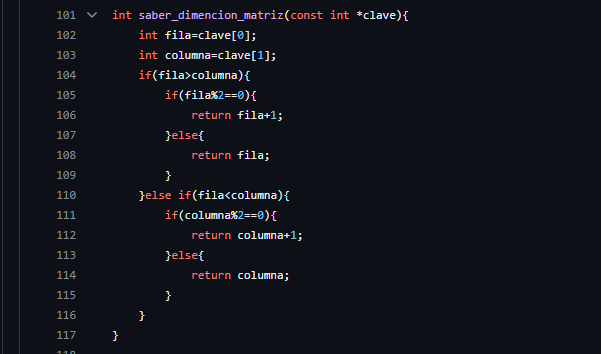
4. Esta función realiza una rotación de 270 grados en sentido antihorario de una matriz cuadrada y luego la imprime en pantalla. La lógica de rotación implica asignar valores especiales a ciertos elementos de la matriz para lograr la rotación adecuada.



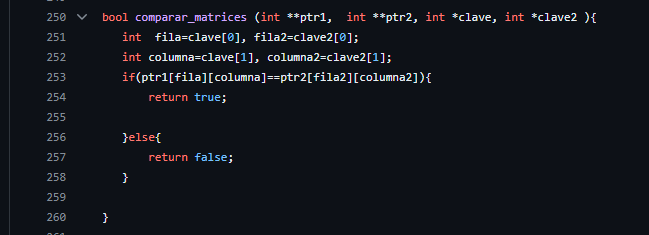
5. La función buscar\_posicion recibe como parámetros un puntero clave que apunta a un arreglo de enteros con la posición de un elemento en una matriz (fila y columna), y un entero siguiente\_matriz que indica si la siguiente matriz es mayor, igual o de menor tamaño. Luego, la función calcula y devuelve un puntero que apunta a un arreglo de enteros con la posición del elemento en la siguiente matriz.



6. saber\_dimencion\_matriz: la función determina la dimensión de la matriz para asegurar que sea lo suficientemente grande para contener tanto la fila como la columna proporcionada, considerando que, si alguno de los valores es par, se incrementa en 1 para asegurar una dimensión impar y tener un punto central en la matriz.

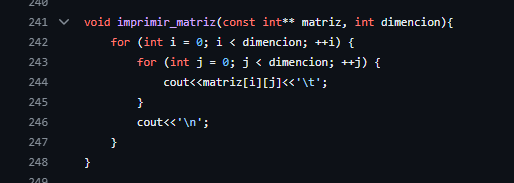


7. la función comparar\_matrices verifica si los elementos en las posiciones especificadas de dos matrices son iguales. Si lo son, devuelve true; de lo contrario, devuelve false.

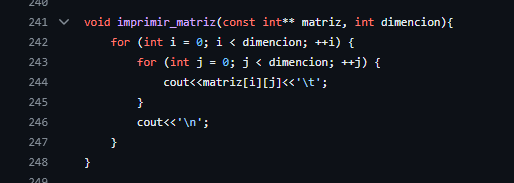


8. la función imprimir\_matriz recorre la matriz y la imprime en la consola de manera ordenada, separando cada elemento por tabuladores y cada fila por saltos de línea.

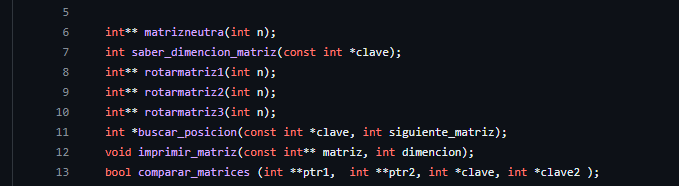
Es importante tener en cuenta que la matriz se trata como una matriz constante en esta función, lo que significa que no se pueden modificar los valores de la matriz dentro de la función imprimir\_matriz. Esto es útil cuando se quiere asegurar que la función de impresión no altere la matriz original.



9. la función liberar\_memoria se encarga de liberar toda la memoria asignada dinámicamente para una matriz creada mediante asignación de memoria dinámica.



10. además de estos puntos también se definieron los prototipos de cada función en la parte superior para facilitar más el desarrollo



## d. Problemas de desarrollo que afrontó.

Durante el desarrollo de nuestro proyecto, nos hemos encontrado con varios desafíos significativos. Uno de los principales ha sido la formulación de la lógica para crear matrices rotadas, Además, hemos experimentado dificultades al utilizar algunas funciones específicas de Git y GitHub. Si bien estas herramientas son fundamentales para el trabajo colaborativo y el control de versiones, su aprendizaje y dominio requieren práctica y comprensión profunda.

A pesar de estos desafíos, consideramos que esta experiencia ha sido muy chévere. La aplicación práctica de los conceptos aprendidos en clase, combinada con la resolución de problemas reales, contribuye significativamente a nuestra formación como futuros ingenieros, de igual manera se estará actualizando el documento de acuerdo con los avances y dificultades que se tengan

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

Nuestro progreso en la resolución del problema ha sido bastante satisfactorio hasta el momento. Tanto mi compañero como yo tenemos una comprensión clara de los conceptos involucrados, lo que nos ha permitido avanzar de manera eficiente en el desarrollo de las funcionalidades requeridas.

En el momento de este commit, la mayoría de las funcionalidades ya están implementadas, lo que indica que estamos progresando a un ritmo sólido.

Además, se estará actualizado el informe con las funcionalidades que se agregaran a lo largo de la evolución del proyecto.

En cuanto a las consideraciones de implementación, es crucial tener un manejo adecuado de las validaciones en nuestro código. Esto implica verificar y asegurar que los datos de entrada sean correctos y estén dentro de los rangos esperados, evitando así posibles errores o comportamientos inesperados.

Asimismo, es importante familiarizarnos y utilizar de manera eficiente las funcionalidades que hemos desarrollado a lo largo de esta semana. Esto incluye aprovechar al máximo las capacidades de las funciones creadas, asegurándonos de su correcto funcionamiento y optimización, nuestro objetivo principal es completar y entregar el desafío antes del tiempo propuesto.