Informe Desarrollo Del Proyecto

Nombres: -Alejandro Zapata Quintero

-Juana María Márquez Guzmán

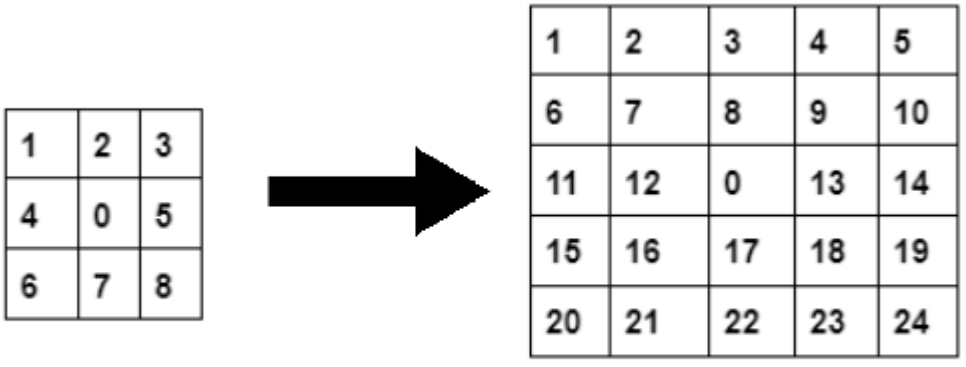
1. Análisis del problema y consideraciones para la alternativa de solución propuesta.

El problema presentado es un desafío de programación que implica el diseño de un sistema de seguridad para la empresa Informa2. Este sistema se basa en el concepto de cerraduras, denominadas X, que tienen características y requisitos específicos. Las cerraduras X están compuestas por varias estructuras M alineadas una tras otra. No hay restricción en la cantidad de estas estructuras y sus tamaños pueden variar.

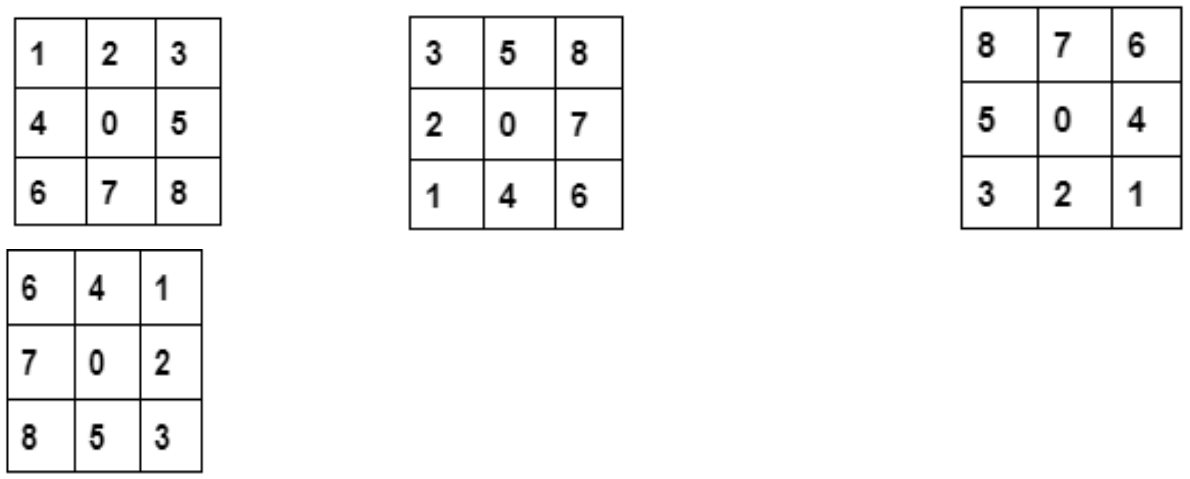
El sistema de apertura de X funciona a partir de la validación de una regla K. Esta regla considera el valor de una celda específica, su posición y la ubicación dentro de las diferentes estructuras alineadas. Para abrir la cerradura, se deben rotar cada una de las estructuras de forma independiente. El objetivo es alinear las celdas de tal manera que la validación de K sea verdadera.

Para resolver este problema, se propuso una solución que implica la creación de una serie de funciones en C + +. Estas funciones permiten:

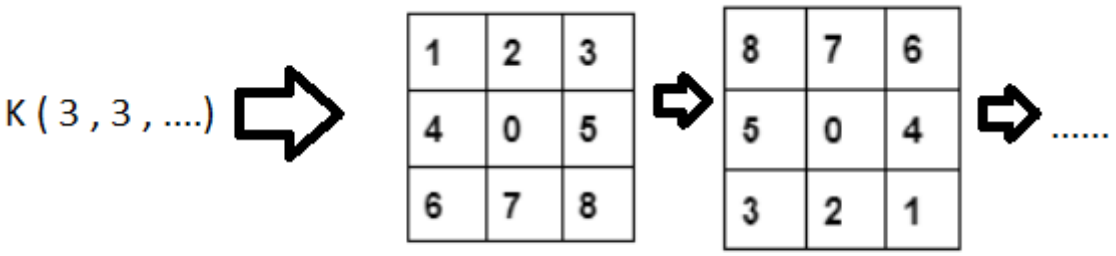
Crear estructuras de datos de tamaño variable: La primera tarea fue diseñar una estructura de datos que pudiera representar las estructuras M de la cerradura X. Estas estructuras M son matrices cuadradas de tamaño impar variable. Para representarlas, se decidió utilizar un arreglo de enteros en C + +. Esta estructura de datos permite crear matrices de cualquier tamaño y acceder a sus elementos de manera eficiente.



Realizar las rotaciones a las estructuras: Una vez que se tuvo una representación para las estructuras M, el siguiente paso fue implementar una función para rotar estas estructuras. La rotación es una operación clave en el funcionamiento de la cerradura X, ya que es la forma en que se alinean las celdas para validar la regla K. La función de rotación toma una estructura M como entrada y devuelve una nueva estructura M que es el resultado de rotar la original 90 grados en sentido antihorario.



Configurar cerraduras de tal forma que la cantidad y el tamaño de las estructuras que la componen sea variable: Con las funciones para crear y rotar estructuras M, se pudo implementar una función para configurar una cerradura X. Esta función toma como entrada una lista de tamaños y crea una cerradura X con una cantidad de estructuras M igual a la longitud de la lista. Cada estructura M tiene un tamaño igual al elemento correspondiente de la lista.



Ejemplo: Variación de llave y cerradura.

Validar una regla de apertura sobre una cerradura: La siguiente tarea fue implementar una función para validar una regla K sobre una cerradura X. Esta función

toma como entrada una cerradura X y una regla K, y devuelve un valor booleano que indica si la regla es válida para la cerradura. Para hacer esto, la función recorre las estructuras M de la cerradura y compara los valores de las celdas especificadas por la regla K.

Generar al menos una configuración de cerradura que se pueda abrir con una regla dada: Finalmente, se implementó una función para generar una configuración de cerradura que se pueda abrir con una regla K dada. Esta función toma como entrada una regla K y devuelve una cerradura X que se puede abrir con esa regla. Para hacer esto, la función crea una cerradura X con una cantidad de estructuras M igual a la longitud de la regla K, y luego rota las estructuras hasta que la regla K sea válida.

Algunos pasos secundarios son:

Validación de entradas: Antes de procesar las estructuras M y las reglas K, es importante validar las entradas para asegurarse de que son válidas. Esto incluye comprobar que los tamaños de las estructuras M son positivos y que las reglas K se refieren a posiciones válidas dentro de las estructuras.

Manejo de errores: Durante la ejecución de las funciones, pueden ocurrir varios tipos de errores, como desbordamientos de memoria o violaciones de segmento. Es importante manejar estos errores de manera adecuada para evitar que el programa se bloquee y para proporcionar mensajes de error útiles al usuario..

Línea de desarrollo y ejecución

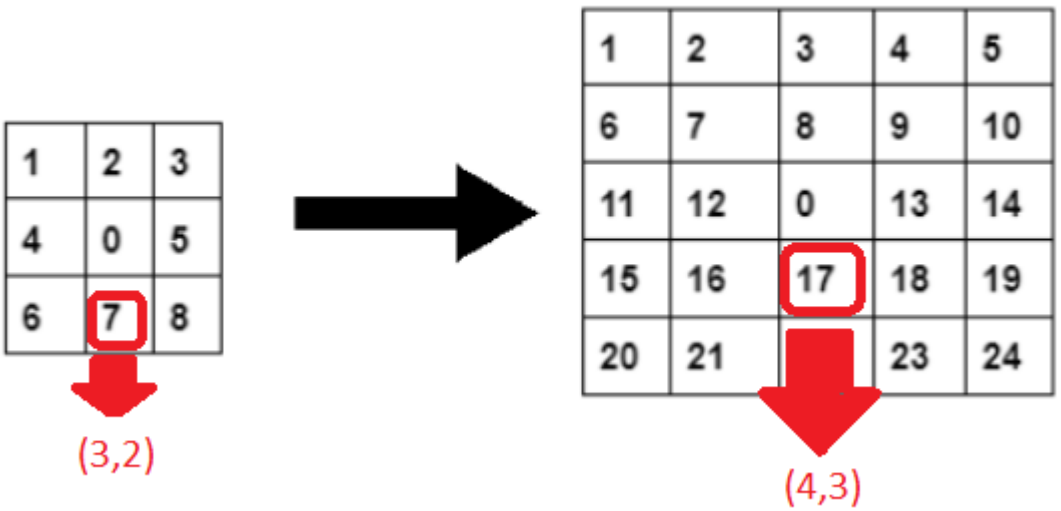
A la hora ya de ejecutar todo este desafío notamos que a la hora de plantear la solución habría que tomar ciertas condiciones y tener claros ciertos conceptos para que el desafío de programación pueda ser abordado con mayor precisión, las soluciones a los distintos problemas que fueron surgiendo son:

se nos dará una llave tal que K(f , c , n,....,n) ,donde la llave tendrá obligatoriamente en sus primeros 2 elementos la posición en fila y columnas respectivamente f y c, esta posición será ubicada en la primera estructura M (Matriz), luego se le ingresaran valores n que pueden tomar valores de -1 , 0 y 1 , estos valores se podrán ingresar sin límite es decir la llave podría variar su tamaño y tener tanto 5 como 10 elementos, Las soluciones que tomamos en este apartado son :

- La llave tendrá que tener por lo menos 3 valores para que funciones el programa, estos valores son fila, columna y por lo menos una comparación

- Para dar el primer valor de la cerradura y de los estados de la estructura M , es decir las rotaciones , se establecieron 2 reglas útiles para un mejor desarrollo, y es que a la hora que el usuario de una llave K , por ejemplo K(5,5,....) podremos notar que esta ubicación se refiere a la fila 5 y columna 5 por lo que la primera matriz tendrá que ser por lo menos de tamaño 5x5 , entonces esta regla servirá tomando el mayor valor impar de estas coordenadas, y en caso de ser par será el mayor valor par + 1 , tal que si nos dan por ejemplo K (2,6….) la estructura M de menor tamaño que podrá contener esta llave será de tamaño 7x7, entonces el programa que haremos no tendrá que procesar matrices con menor tamaño al posible ya que en este caso no tiene sentido que después de este arreglo halla una matriz en la cerradura de menor tamaño que 7x7 y como este primer valor no tiene que cumplir ninguna comparación estará en estado neutro por definición.

-teniendo como base la menor estructura y pensando que estas matrices están alineadas desde el centro , se puede ver que para hallar las distintas comparaciones además de rotar se tendrá que aumentar la dimensión de las matrices , notamos que entre cada 2 dimensiones que suba una matriz , la posición inicial subirá 1 en fila y 1 en columna , visto desde un ejemplo si nos dan una llave cuyos primeros 2 valores sean K(3,2,...) notamos que tiene una posición 3,2 en una matriz 3x3 , sin embargo en la matriz 5x5 esa posición aumentaron 1 en columna y 1 en suma es decir sería igual a 4,3, y así sucesivamente para las distintas matrices.



-Teniendo en cuenta todo lo anterior, logramos crear un programa funcional que, dada una regla ‘k’ proporcionada por el usuario, seleccionaba el número impar más grande de los dos primeros números, que corresponden a la fila y la columna. En caso de ser par, se le sumaba uno. Por lo tanto, la primera matriz que el programa tomaba era del tamaño hallado anteriormente y se encontraba en el estado neutro (el cual no tiene ninguna rotación). Este sería el primer elemento de la cerradura con su respectivo estado. Este dato se iba guardando en los arreglos ‘cerradura’ y ‘rotaciones’, ambos utilizados en el código.

Después de esto, se tomaban en cuenta las comparaciones ingresadas por el usuario. Si ‘comp’ es -1, verifica si ‘anterior’ es menor que ‘próximo’. Si ‘comp’ es 0, verifica si ‘anterior’ es igual a ‘próximo’. Si ‘comp’ es 1, verifica si ‘anterior’ es mayor que ‘próximo’. Teniendo en cuenta esto, para el siguiente elemento de la cerradura se rotaba la matriz del tamaño definido anteriormente, la cual es la menor que puede tener la cerradura, y se rotaba a los diferentes estados buscando un número que cumpliera la condición de la comparación. En caso de no encontrarlo en esa matriz, se le aumentaba dos a la dimensión de la misma (aumentando en 1 también su fila y columna para mantenerse alineados con el 0) y también se hacían las respectivas rotaciones a los respectivos estados. Así sucesivamente hasta encontrar un número que cumpliera con la condición de la comparación y lo agregaba al respectivo arreglo.

Problemas de desarrollo que afrontó

Durante el proceso de desarrollo, se enfrentaron varios desafíos que requirieron atención y resolución cuidadosa. Algunos de los problemas más significativos incluyeron:

- Validación de entradas: Garantizar que las entradas proporcionadas por el usuario fueran válidas y cumplieran con los requisitos del sistema fue un desafío importante. Se implementaron mecanismos de validación para asegurar la integridad de los datos ingresados.

En cuanto a la validación de entradas, el problema principal, que consideramos un reto, fue el caso en el que el usuario ingresa una cierta cantidad de ‘1’ seguidos. Esto causaba que el programa se quedara cargando sin arrojar jamás un resultado. Era evidente que esto sucedería, ya que el programa, dependiendo del número que se esté comparando, solo podía soportar cierta cantidad de rotaciones en las que se buscara que este número fuese mayor. Por ejemplo, la gran mayoría de los números antes del cero en la estructura (matriz) no podían soportar ni siquiera dos ‘1’ seguidos, y los números después del cero solo podían soportar dos, algunos tres en el caso de tener antes un ‘-1’.

Por lo tanto, la mejor solución para este problema fue un condicional que contará la cantidad de iteraciones desde la última para evitar ambigüedades. Cuando el programa pasaba cierta cantidad de iteraciones, significaba que se había producido alguno de estos errores en los que el programa no arrojaría un resultado. Por lo cual, en la pantalla saltará un mensaje de ‘valor inválido’.

Además, en cuanto a las validaciones, se agregaron las respectivas excepciones en caso de agregar un tipo de dato que no es el pedido por el programa, o un dato que no se encuentre en el rango de números pedidos.

- Optimización del rendimiento: La eficiencia del algoritmo fue una preocupación constante, especialmente en operaciones que involucran manipulación de matrices y rotaciones. Se realizaron esfuerzos para optimizar el código y mejorar el rendimiento del programa.

- Manejo de errores: Identificar y manejar adecuadamente posibles errores durante la ejecución del programa fue crucial. Se implementaron estrategias para detectar y resolver errores, como fugas de memoria o violaciones de segmento, garantizando así la estabilidad y robustez del sistema.

- Complejidad del algoritmo: La naturaleza compleja del algoritmo requería una comprensión profunda del problema y una cuidadosa planificación de la solución. Se dedicó tiempo a analizar y diseñar un enfoque efectivo para abordar el desafío de programación.

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

La evolución de la solución propuesta pasó por varias etapas esenciales, desde la representación de datos hasta la generación y validación de configuraciones de cerradura. Se reconoció la necesidad de una estructura de datos que pudiera manejar las múltiples características de las cerraduras x, lo que resultó en un sistema adaptable basado en matrices cuadradas de tamaño variable.

El enfoque de la implementación fue desarrollar funciones en C++ para la creación, rotación y validación de las estructuras M, así como para la configuración de las cerraduras x y la generación de configuraciones válidas para una regla k específica. Estas funciones fueron diseñadas para ser eficientes y flexibles, capaces de ajustarse a diferentes escenarios y requisitos de entrada.

Durante el desarrollo, se puso especial énfasis en la validación de las entradas para asegurar la integridad de los datos procesados. Se establecieron mecanismos de verificación para confirmar que los tamaños de las estructuras M fueran correctos y que las reglas k se refirieran a posiciones válidas dentro de las matrices. Además, se implementaron medidas de manejo de errores para gestionar posibles fallos durante la ejecución del programa y proporcionar retroalimentación clara al usuario en caso de problemas.

A medida que se progresaba en la implementación, surgieron consideraciones importantes relacionadas con la lógica operativa de las cerraduras x y las reglas de apertura asociadas. Se definieron reglas específicas para la configuración inicial de las cerraduras y las rotaciones de las estructuras M, lo que permitió optimizar el proceso y asegurar la consistencia en los resultados obtenidos.

En resumen, la evolución de la solución abordó una serie de desafíos técnicos y conceptuales, desde la definición de la estructura de datos hasta la implementación de funciones especializadas para manipular y validar cerraduras x. Las consideraciones para la implementación se centraron en garantizar la robustez, la eficiencia y la usabilidad del sistema, lo que se logró mediante un enfoque cuidadoso en cada etapa del desarrollo.