**INFORME DESAFIO #1**

Juan José Quiceno Pabón

Programa de Ingeniería en Telecomunicaciones

Universidad de Antioquia

Facultad de Ingeniería

Augusto Salazar

Aníbal Guerra

Medellín, Colombia

2024

**ANÁLISIS DE POSIBLES SOLUCIONES**

**//**A primera vista, cual es la forma en que usted abordaría el problema?

Como primeras observaciones al desafío planteado, vemos un problema que nos solicita el revisar los elementos contenidos dentro de varias estructuras, los cuales una vez referenciados se procederá a realizar validaciones con elementos de otros conjuntos del sistema alineado, para así permitir o denegar la apertura de una cerradura, teniendo el denegar como consecuencia, la rotación de la estructura de datos hasta finalmente dar con un valor permitido.

En cuanto a la parte de codificación, se podría decir que es clara la necesidad de implementar arreglos dinámicos que contengan la información de cada estructura de datos, el uso de funciones que me permitan construir, rotar, modificar y validar cada matriz.

**Otras posibles implementaciones:**

* Se podría usar un arreglo que contenga cada estructura de datos, lo que nos lleva a un posible triple puntero, uno que apunte a cada estructura, y los otros dos que apuntan a cada estructura de datos
* Una función que permita el ingreso manual de la regla K

Personalmente creo que la mayor dificultad que podemos encontrar en el problema es las partes relacionadas con la regla K, ya que no es muy claro el mensaje que me da tras un primer análisis, este me deja la idea de un arreglo el cual apunta a n matriz, en x posición, en la cual su valor debe ser menor o mayor al de la siguiente matriz, y este también debe ser menor o mayor al siguiente, dependiendo de la secuencia determinada por los valores de las ultimas 2 celdas de k, lo cual conlleva ciertas rotaciones para que esta condición se cumpla.

Es importante resaltar que este es un análisis “En crudo” del problema planteado, por lo que todavía no se define bien el cómo se abordará la solución del problema.

**ANALISIS DEFINITIVO**

//Que entendio usted del problema una vez terminado

**METODOLOGIA**

**//**Cual será la forma en que se abordará el problema?

En vista de que el problema abordará lo visto en la primera parte del curso y evaluará los conceptos aprendidos en este, el código estará dividido en diferentes módulos, dentro de ellos el módulo principal, que es donde se realizarán las respectivas invocaciones de distintas funciones según corresponda; Otros tendrán como contenido diferentes funciones en base a su aplicación en el código, esto para no saturar uno o varios módulos con líneas de código.

Las partes mas importantes del programa se desarrollarán de ultimo, para así ir construyendo poco a poco la arquitectura en base a las funciones mas “sencillas”, ya que pensar inmediatamente en, por ejemplo, la función que comprueba la regla K para la apertura de las estructuras de datos X y que según dicte la necesidad ir desarrollando las funciones necesarias como la rotación o generar la regla K, seria una muy mala práctica a mi parecer.

El informe irá creciendo conforme el progreso en el análisis y la escritura de código avance, de este modo se añadirán exclusivamente ideas claras de lo realizado a excepción del análisis sobre posibles soluciones.

**FUNCIONES UTILIZADAS**

Se especificará mas a fondo las funciones de mayor complejidad a la hora de su desarrollo:

* **Locked**:Esta encargada de comprobar la regla K y generar una configuración para la apertura de las estructuras X.
  + **Variables**
* **Rotation**:Esta encargada de rotar una estructura de datos especifica cuando sea necesario, esta rotara 90° la matriz cuantas veces sea necesaria.
  + **Variables**
    - **orden**: Variable de tipo entero, la cual se inicializa con el valor que se encuentre en la posición matriz (Parámetro de la función) del arreglo pref, el cual guarda el orden de cada una de las estructuras de datos.
    - **backup y backup\_2**: Variables de tipo entero, estas se inicializan con valor 0, y conforme los ciclos dentro de la función iteran, estas van cambiando su valor. Su función es similar, ser el apoyo para las posiciones en la matriz que serán reemplazadas, para posteriormente moverlas a otra posición.
    - **help**: Variable de tipo entero que se inicializa con valor 0, conforme avanzan los ciclos esta toma los valores de i (variable del primer ciclo) cuando sale del segundo ciclo, en el cual aumenta de 1 en 1, su funcionamiento es clave dentro de la lógica de **Rotation**, ya que conforme j avanza (variable que referencia cada columna), help sirve como un “suplente” de i en consecuencia a la necesidad de j de no volver a tomar el primer valor dentro del ciclo, ya que volvería a rotar algo que no debe, entonces help lo ayuda a posicionarse en el siguiente elemento a rotar.
    - **cont**: Variable de tipo entero que se inicializa con valor 0, esta le sirve a j para que su rango de acción dentro de la matriz, disminuya conforme avanza. Por ejemplo, cuando j termina con los bordes de la matriz, necesita “adentrarse” mas, por lo que no puede tomar valores relacionados con los bordes, aunque siempre empiece desde ellos, cont se resta a su valor inicial y lo ayuda a posicionarse cuando entra al ciclo nuevamente.
* **User\_Bob:** Se encarga de construir las matrices a gusto del usuario, más específicamente la matriz de la estructura de datos y la otra se guarda el orden de cada una de las estructuras
  + **Variables:** 
    - **mat:** Aunque este declarada en el ámbito global como un triple puntero, mat se inicializa dentro de esta función como un arreglo de arreglos, teniendo en su primer nivel un espacio indicado por el usuario según el numero de estructuras que desee, un segundo nivel que tiene el numero de filas de cada matriz según el orden ingresado, y un tercer nivel donde se agregan n elementos a esa fila.
    - **pref**: Se declara en el ámbito global pero se inicializa aquí, su trabajo es guardar el orden de cada estructura de datos en un arreglo simple, el cual esta intencionalmente alineado con cada estructura según su posición.
    - **est y orden**: Variables de tipo unsigned short int, esto debido a que no precisamos de números negativos ni tampoco muy grandes. Se encargan de guardar el valor ingresado por consola.
* **Rule\_K**:Genera la regla K a gusto del usuario.
* **Comp:** Su función es validar algunas entradas relacionadas con rangos.
* **Value:** Su función consta en asignarle los valores a cada posición de la matriz que reciba en el parámetro.
* **Graph**:La finalidad de esta función es imprimir la estructura de datos que se le indique en el parámetro, su accionar dentro del programa es meramente gráfico.
* **Eraser:** Su función es liberar la memoria reservada por cada uno de los punteros dinámicos al final del programa.

**OBSERVACIONES**

🡪Dudas acerca del tamaño variable de cada una de las matrices, como serán las funciones que nos ayudaran con las validaciones, como haremos las validaciones?, Rotacion Independiente?, Cual es el escenario en que la matriz cambia de tamaño y rota?.

🡪Avances día 1:

* Función para construir, imprimir, rotar, borrar e inicializar arreglos.
* Gracias al aporte del tutor se logro concretar una estrategia con el triple puntero, ya que se estaba pensando en meter en un mismo arreglo todos los datos o también independizar cada uno de las estructuras de datos. a raíz de estos consejos se creo otro arreglo que esta alineado intencionalmente según la posición de la matriz, en el cual se guarda el respectivo orden de cada una debido a su alta demanda dentro del código.

🡪Avances día 2:

* Se cambio el tipo de dato de algunas variables ya que no precisaban de tanto espacio en memoria
* Se añadió una pequeña validación de entrada en la parte que el usuario ingresaba el orden de cada matriz
* Se agrego la función para construir la regla K, al igual que otra función que sirve como validación de entrada para ciertos casos relacionados con rangos

🡪Avances día 3:

* Se desarrollo la función que configura la apertura de la cerradura según la regla K, todavía se siguen teniendo dudas acerca de la secuencia de apertura, ya que pueden ocurrir ciertas coincidencias respecto a la posición “vacía” con la que se alinean las estructuras, ¿Se omite? ¿Las filas y columnas deben ser proporcionales en función al orden de la matriz?, esas y otras preguntas han surgido durante el desarrollo, de momento se ha configurado para matrices del mismo orden y para una secuencia mayor / menor o menor / mayor.
* PROBLEMA: Al pensar que la secuencia la podía determinar el usuario, surgía un problema en cuanto a las rotaciones, ya que al buscar un número más grande que el de mayor valor en la estructura entraba en un bucle infinito, por lo que la secuencia debía ser intercalada según el 1 o -1.