**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 INICIAL**

Como primeras observaciones al desafío planteado, vemos un problema que nos solicita revisar los elementos contenidos dentro de varias estructuras. Una vez referenciados, procederemos a realizar validaciones con elementos de otros conjuntos del sistema alineado, para así permitir o denegar la apertura de una cerradura. Denegarla implicará 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. Además, se requerirá el uso de funciones que permitan construir, rotar, modificar y validar cada matriz.

**Otras posibles implementaciones:**

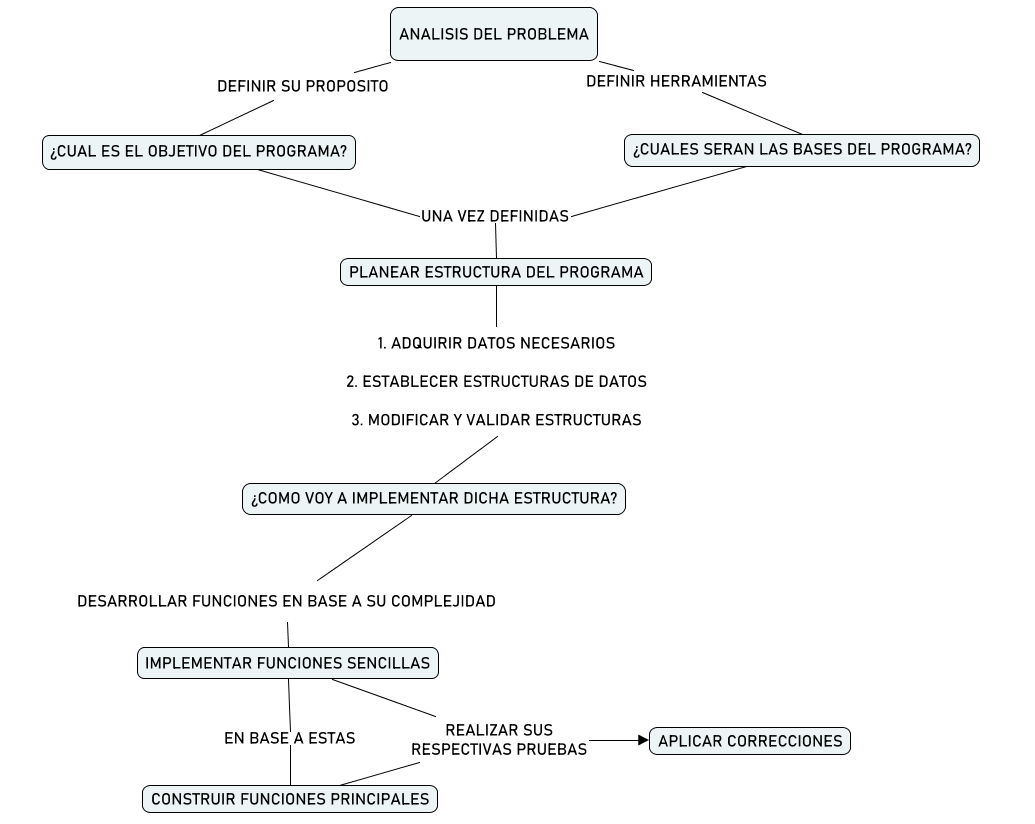
* Se podría usar un arreglo que contenga cada estructura de datos, lo que nos lleva a un posible triple puntero, los 3 niveles constan de un arreglo que contenga n estructuras, otro que tenga n espacios dentro de cada estructura y uno más donde se tenga m espacios dentro de cada n.
* Una función que permita el ingreso manual de la regla K.

Personalmente, creo que la mayor dificultad que podemos encontrar en el problema está relacionada con la regla K. Tras un primer análisis, no queda muy claro el mensaje que transmite. Parece implicar un arreglo que apunta a n matrices, donde en cada posición x, el valor debe ser menor o mayor que el de la siguiente matriz, y a su vez, este valor debe ser menor o mayor que el siguiente, dependiendo de la secuencia determinada por los valores de las últimas 2 celdas de K. Esto 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.

**METODOLOGIA**

En vista de que el problema abordará los conceptos adquiridos en la primera parte del curso y evaluará los aprendidos en este, el código estará dividido en diferentes módulos. En ellos, el módulo principal será donde se realicen las respectivas invocaciones de diversas funciones según corresponda. Otro contendrá diferentes funciones, esto con el fin de evitar saturar uno o varios módulos con muchas líneas de código.

Las partes más importantes del programa se desarrollarán al final, para así construir gradualmente unas bases a partir de funciones más "sencillas". Esto implica pensar primero en funciones básicas y luego, según la necesidad, desarrollar las más complejas, como la que comprueba la regla K para la apertura de las estructuras de datos X, o funciones para rotación y generación de la regla K. Esto evitará prácticas poco eficientes y favorecerá un desarrollo ordenado y fraccionado del programa.

El informe irá creciendo conforme avance el progreso en el análisis y la escritura de código. De este modo, se añadirán exclusivamente ideas claras de lo realizado, a excepción del análisis sobre posibles soluciones. Además, se destinará una parte del informe para mostrar los avances diarios a modo de bitácora. El esquema mostrado previamente representa lo mencionado.

**ALGORITMOS IMPLEMENTADOS**

En base a un análisis previo del problema, se establecieron ciertos parámetros para desarrollar el código con el fin de facilitar su construcción. La mayoría de las funciones tienen un área muy específica dentro de la cual operan. Por ejemplo, gran parte de las que están descritas a continuación no reciben un arreglo que contenga varios elementos como parámetro, sino que reciben un único elemento cada vez que se necesite. Esto se hace para evitar abordar numerosos escenarios en caso de errores de semántica. Otro parámetro clave fue definir su área de trabajo, dividiéndose en "Constructoras" y "Operadoras". Donde las constructoras crean las bases de datos que las operadoras manipulan. A continuación, se presentan las funciones resultantes:

**Funciones Constructoras:**

* **User\_Bob:** Se encarga de construir las matrices a gusto del usuario en 2 de los punteros declarados (mat y pref), dentro de mat de una habrá un arreglo de 3 niveles donde se guardarán cada una de las estructuras solicitadas con su respectivo orden contenido en pref, esto con el fin de no abusar de la estructura de datos principal a raíz de la alta demanda del orden de cada matriz dentro del código.
* **Value:** Su función consta en asignarle los valores a cada posición de la matriz que reciba en el parámetro en base a su orden.
* **Rule\_K**:Genera la regla K a gusto del usuario.

**Funciones Operadoras:**

* **Locked**:Se podría decir que es la función más importante dentro del programa ya que aborda todos los algoritmos desarrollados previamente a ella, esta realiza las respectivas validaciones para cumplir con la regla K en base a un concepto de proporción que me ayuda a alinear por medio de las coordenadas cada matriz.
* **Rotation**:Esta encargada de rotar una estructura de datos especifica cuando sea necesario, esta rotara 90° la matriz cuantas veces sea invocada con la posición de la matriz como argumento.Para su lógica me base mas que todo en los patrones de las posiciones y no de valores, ya que siempre va a ser un ciclo de 4 movimientos entre posiciones relacionadas con las esquinas de la matriz.
* **Comp:** Su función es validar algunas entradas relacionadas con rangos de 1 a n.
* **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 al final del programa.

**CONCLUSIONES**

Mi análisis del desafío, después de casi 1 semana completa de trabajo y estar en su fase final, es que nos solicita un numero de estructuras M, las cuales estarán alineadas por su centro gracias a ciertas proporciones en relación a sus coordenadas. Estas serán manipuladas por ciertas funciones en base a una regla K, que indica una secuencia mayor – menor o viceversa a partir de una estructura M seleccionada, provocando así cambios en su tamaño o rotaciones de algunas matrices para cumplir con las condiciones impuestas por K, permitiendo así, generar una configuración de apertura en la que todas las condiciones se cumplen.

Conforme progresaba el trabajo, mi forma de ver el problema cambio constantemente al visualizar los diferentes escenarios por los que podría pasar el programa durante la ejecución del código, sembrando constantemente dudas, las cuales serian tenidas o no en cuenta provocando cambios dentro del algoritmo. Funciones como **Dimension** son un ejemplo claro de ello, ya que esta fue tenida en cuenta al principio a raíz del análisis inicial del problema, pero a medida que avanzaba y me asesoraba con el monitor, decidí hacerla a un lado temporalmente a consecuencia de un análisis que podría decirse fue erróneo, más sin embargo no lo es, ya que puede tratarse de una de las muchas posibles soluciones al desafío, donde puede haber una combinación de estructuras de datos en la cual sea imposible cumplir con las reglas generadas por K, sin importar cuantas veces rote las matrices.

La planificación inicial fue fundamental al momento de codificar, ya que el orden seguido durante el desarrollo estuvo estrechamente ligado a la planificación. Por ejemplo, al final el segmento de código al que más dedicación le otorgué fue la función "Locked", tal como había previsto. Conceptos como proporción y cambio de tamaño para matrices en ciertos escenarios resultaron clave para que cualquier matriz ingresada pudiera adaptarse a la configuración de apertura. Estas y otras funciones, desde mi punto de vista, conformaron un algoritmo capaz de buscar siempre una solución ante cualquier escenario.

**AVANCES**

* **Día 1**:
* Función para construir, imprimir, rotar, borrar e inicializar arreglos.
* Gracias a la ayuda del tutor se logró 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 creó 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.
* **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
* **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.
* **Día 4**:
* Se corrigió la función **Rotation** ya que no funcionaba con matrices de 6x6 en adelante a partir de un concepto de “capas”, también se añadieron muchos condicionales a la función **Locked**, esto debido a la alta cantidad de escenarios en los que la posición seleccionada debe adaptarse a una matriz de mayor orden.
* **PROBLEMA**: No siempre la matriz que debe rotar es la de adelante, ya que se puede dar el caso en que rotándola por completo no se puede obtener un número mayor al apuntado de la matriz referenciada, por lo que se deben buscar soluciones para este posible escenario, ¿Aumentar la matriz de tamaño? ¿Rotar la matriz actual?
* **PROBLEMA**: Durante las pruebas y correcciones que se le realizaban a **Locked** (Función para determinar la apertura de X), se determinó que la función **Rotation** tenía ciertos problemas con matrices de cierto orden.
* Se eliminaron y añadieron algunas variables de **Rotation** a raíz de su corrección.
* Se determino un límite para las rotaciones de la matriz, esto debido a que al rotar 3 veces, todos sus valores fueron evaluados. En caso de que la otra matriz con la que se evaluó también realice esas 3 rotaciones, se da por hecho que no es posible cumplir con las condiciones que K me pide.
* Una posible solución al problema en que no es suficiente con rotar las matrices, es la de aumentar su tamaño hasta cumplir con las condiciones
* **Día 5**:
* Se añadió finalmente la función **Dimension**, esto debido a que surgieron ciertas confusiones al momento de analizar la parte donde M puede tener tamaño variable, no se refería solamente al momento de inicializarla, sino que también al escenario en que M podría no cumplir con las condiciones, lo que provocaría un crecimiento en su orden para lograrlo.
* Se realizaron modificaciones a **Locked** en cuanto a su lógica y el anticipo a ciertos escenarios, por ejemplo, donde una matriz gira por completo y no se cumple la condición, la otra es la que va a girar, y si se vuelve a repetir este escenario, la matriz que esta siendo referenciada aumenta de tamaño.
* **PROBLEMA**: Después de numerosas pruebas, se determinó que hay algún problema en cuanto a la segmentación cuando la primera matriz de la estructura es más grande que la segunda.
* Se lograron solucionar algunos problemas relacionados con la lógica del programa, uno relacionado al momento en que la matriz cambiaba su tamaño y la proporción que llevaba desde el principio ligado a las coordenadas, y otro donde dependiendo la fase de secuencia (1 o -1) se debía aumentar el tamaño de la matriz referenciada o la matriz con la cual se comparaba.
* Se destino un tiempo a comentar unas cuantas líneas de código, no se hace solo como una obligación, sino con el objetivo de afirmar la idea que tuve a la hora de desarrollar ciertos segmentos de código.
* **Día 6**:
* **PROBLEMA**: Cuando digito una coordenada en una matriz 7x7 muy grande, por ejemplo, fila 6 y columna 3, cuando se hace la proporción con una matriz 3x3 se da un problema de segmentación.
* **Día 7**:
* Se trabajó exclusivamente en el informe.
* **Día 8**:
* Se añadieron 2 validaciones dentro del ciclo que compara valores, ahora cuando las coordenadas digitadas en K superen el orden de alguna matriz aumentará su tamaño, y la otra, en caso de que las proporciones den un número negativo, provocando un error de segmentación, solo aplicara proporciones a ciertas coordenadas o en dado caso, a las 2.
* **PROBLEMA:** ¿Siempre debe empezar desde la 1era estructura?, y en ese caso, ¿Qué validaciones debo realizar para que se cumplan las condiciones desde n matriz?
* **Día 9**:
* La variable rep paso de string a char, ya que era innecesario incluir una librería solo para un pequeño segmento de código, a cambio de eso solo puede recibir el primer digito de lo que digite el usuario.
* Se modifico la parte de la secuencia, ahora el usuario podrá elegir a su gusto la secuencia entre matrices, no solo será intercalada sino que también podrá, por ejemplo, poner menor – menor.
* **Día 10:**
* Se corrigió el problema detectado en el día 8, fue un cambio muy pequeño, pero creo yo es de vital importancia.
* Se añadieron algunas cosas al informe, para así darlo por finalizado.
* **Día 11**:
* Sin cambios.
* **Día 12**:
* Se dividió el modulo de funciones en 2, para que así tuviera sentido lo descrito en el informe(Constructoras y operadoras).