Desafío 1

Juan Diego Arias Toro

C.C.1192792031

Informática II

2024-1

Universidad de Antioquia

Fac. Ingeniería Electrónica y Telecomunicaciones

Análisis del problema y consideraciones para la alternativa de solución propuesta

El problema planteado en el desafío consiste en desarrollar un sistema capaz de generar una configuración de cerradura "X" que satisfaga una regla de apertura "K" dada. La cerradura "X" se compone de varias estructuras de datos "M" alineadas, cada una de las cuales puede rotar y cambiar de tamaño, con la restricción de que el número de filas y columnas debe ser impar e igual.

Para abordar este problema, se han considerado las siguientes alternativas de solución:

1. Representación de las estructuras M: Se ha optado por utilizar una clase StructureM para representar cada una de las estructuras M que componen la cerradura X. Esta clase permite crear, rotar y obtener el valor de las celdas de la estructura.

2. Configuración de la cerradura X: La clase LockX se encarga de administrar el conjunto de estructuras M que forman la cerradura X, permitiendo validar una regla K y generar una configuración que satisfaga dicha regla.

3. Generación de la configuración: La función generateConfiguration() en la clase LockX se encarga de encontrar una configuración válida de la cerradura X rotando cada una de las estructuras M de forma independiente hasta que se cumple la regla K.

Estas consideraciones han permitido diseñar una solución modular y extensible, donde cada componente tiene responsabilidades bien definidas, facilitando el desarrollo, mantenimiento y posibles futuras ampliaciones del sistema.

Esquema de tareas

1. Creación de Estructuras de Datos de Tamaño Variable
    - Definir la estructura de datos M que puede girar y cambiar de dimensiones.
    - Implementar la capacidad de crear estructuras de datos con tamaños variables, siguiendo las Figuras 1 y 2 del documento.

2. Implementación de Rotaciones en las Estructuras

- Desarrollar funciones para realizar rotaciones en las estructuras de datos M, considerando los estados de rotación 1, 2 y 3.
- Asegurar que las rotaciones se realicen de manera correcta y coherente para alinear las celdas según las reglas de apertura.

3. Configuración de Cerraduras X con Tamaños Variables
   - Crear un módulo que permita configurar cerraduras X con diferentes tamaños y alineaciones de estructuras M.
   - Alinear varias estructuras M una tras otra, utilizando la celda del centro como referencia, sin restricción en la cantidad de estructuras ni en sus tamaños.

4. Validación de Reglas de Apertura sobre las Cerraduras X
   - Implementar funciones para validar reglas de apertura sobre las cerraduras X, considerando la regla K y las condiciones específicas de cada cerradura.
   - Garantizar que la validación de las reglas sea precisa y eficiente para determinar la apertura de la cerradura.

5. Generación de Configuraciones de Cerraduras X
   - Desarrollar un módulo que genere configuraciones de cerraduras X que puedan abrirse con una regla K dada.
   - Hay que asegurar que las configuraciones generadas cumplan con las reglas de apertura establecidas y puedan ser abiertas correctamente.

Algoritmos implementados

Creación de la matriz: El algoritmo de creación de la matriz M consiste en rellenar una matriz cuadrada impar con valores secuenciales, colocando un 0 en el centro.

Rotación de la matriz: El algoritmo de rotación de la matriz M consiste en crear una nueva matriz temporal y copiar los valores de las celdas en la nueva posición, de acuerdo con la rotación deseada.

Creación de la cerradura X: El algoritmo de creación de la cerradura X consiste en crear múltiples estructuras M y almacenarlas en un arreglo tridimensional.

Extracción de valores: El algoritmo de extracción de valores consiste en convertir las coordenadas cartesianas de una celda a la posición correspondiente en el arreglo que representa la cerradura X, y extraer el valor de dicha celda.

Validación de la regla K: El algoritmo de validación de la regla K consiste en recorrer los valores de las celdas alineadas y verificar que se cumplan las condiciones impuestas por la regla.
Validación de relaciones: El algoritmo de validación de relaciones consiste en comparar los valores de las celdas alineadas de acuerdo con las relaciones especificadas en la regla K.

Generación de la configuración: El algoritmo de generación de la configuración de la cerradura X consiste en rotar cada una de las estructuras M hasta que se cumple la regla K para esa estructura.

Estos algoritmos se han implementado en las diferentes funciones del código, siguiendo una estructura modular y reutilizable.

Problemas de desarrollo y evolución de la solución

Durante el desarrollo de la solución, se han enfrentado algunos retos, principalmente relacionados con la manipulación de las estructuras M y la validación de la regla K. Inicialmente, se optó por utilizar una estructura de datos simple para representar las celdas de la estructura M, pero se encontró que esto dificultaba la implementación de las funciones de rotación. Por lo tanto, se decidió utilizar una estructura Cell que contiene los valores, filas y columnas de cada celda, lo cual simplificó el código de rotación.
Otro desafío fue la implementación de la validación de la regla K. Inicialmente, se intentó utilizar una sola función que validara toda la regla, pero esto resultó ser complejo y difícil de mantener. Por lo tanto, se optó por dividir la validación en pasos más pequeños, lo cual mejoró la legibilidad y modularidad del código.

Consideraciones finales

La solución propuesta cumple con los requisitos especificados en el desafío y permite generar configuraciones de cerradura X que satisfacen una regla K dada. La arquitectura modular y el uso de estructuras de datos apropiadas han sido fundamentales para lograr una implementación robusta y extensible.
Algunas consideraciones finales a tener en cuenta:

- El programa es capaz de manejar cerraduras X de tamaño y cantidad de estructuras variable, lo cual lo hace flexible y adaptable a diferentes escenarios.

- La separación de responsabilidades entre las clases StructureM y LockX facilita el mantenimiento y posibles futuras ampliaciones del sistema.

- La implementación utiliza punteros, arreglos y memoria dinámica, cumpliendo con los requisitos mínimos establecidos.

En resumen, la solución propuesta aborda de manera satisfactoria el desafío planteado, demostrando las habilidades en análisis de problemas y programación en C++ del desarrollador.