Informe Final Parcial II

Nombre: -Alejandro Zapata Quintero

-Juana María Márquez Guzmán

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

Contexto del Sistema de Metro:

Este informe detalla la creación y desarrollo de un simulador de red de Metro programado en C++. El objetivo principal de este proyecto es generar una herramienta interactiva que facilite a los usuarios el estudio y entendimiento de la operación de un sistema de transporte, como el Metro, en una ciudad imaginaria.

Evaluación del Problema: El reto propuesto implica la creación y desarrollo de un sistema que pueda simular la complicada red de Metro de una ciudad. Esto conlleva la generación de una estructura de datos que represente de manera adecuada las estaciones, las líneas y la red en su totalidad. Adicionalmente, el sistema debe permitir la adición de estaciones a las líneas, llevar a cabo simulaciones de tiempos de llegada entre estaciones y ofrecer otras funcionalidades cruciales para la administración eficaz del Metro.

Selección de Estructuras de Datos: Se realizó una cuidadosa selección de las estructuras de datos más adecuadas para almacenar la información de las estaciones, líneas y la red del Metro. Se optó por utilizar arreglos y punteros normales debido a su eficiencia en el manejo de datos y su bajo consumo de recursos. Esta elección garantiza un rendimiento óptimo del simulador, incluso cuando se trabaja con grandes cantidades de información. Desafíos en la Gestión de Líneas y Estaciones:

Dinamismo de la Red: Las redes de metro están sujetas a cambios constantes, como la adición de nuevas estaciones, la extensión de líneas existentes o la creación de nuevas líneas. La solución debe ser lo suficientemente flexible como para adaptarse a estos cambios sin comprometer la integridad del sistema.

Criterio de eficiencia: La eficiencia en la operación del metro es esencial para minimizar los tiempos de espera y maximizar la capacidad de transporte. La solución debe permitir una gestión eficiente de las líneas y estaciones para optimizar la experiencia del usuario.

Seguridad y Fiabilidad: La seguridad y la fiabilidad son aspectos críticos en la operación del metro. La solución propuesta debe garantizar la integridad de los datos y la estabilidad del sistema.

Interconexión de Líneas: Las estaciones de transferencia entre líneas son puntos clave en la red de metro. La solución debe ser capaz de identificar y gestionar estas estaciones de manera efectiva para facilitar la transferencia de pasajeros entre líneas.

Consideraciones para la Solución Propuesta:

Modelado Orientado a Objetos: El enfoque orientado a objetos proporciona una representación natural de las entidades del sistema, como las estaciones y las líneas. Esto permite una abstracción efectiva de la complejidad del sistema y facilita su comprensión y mantenimiento.

Gestión Dinámica de Datos: Dado el dinamismo inherente de las redes de metro, la solución debe ser capaz de gestionar de manera eficiente la adición, eliminación y modificación de líneas y estaciones. El uso de estructuras de datos dinámicas, como punteros y listas enlazadas, puede ser beneficioso para facilitar esta gestión.

Optimización del Rendimiento: La solución debe ser capaz de manejar grandes volúmenes de datos y realizar operaciones complejas de manera eficiente. Se deben implementar algoritmos y estructuras de datos eficientes para garantizar un rendimiento óptimo del sistema, incluso en escenarios de carga pesada.

Seguridad y Robustez: Se deben incorporar mecanismos de seguridad y validación de datos para proteger el sistema contra posibles ataques y garantizar la integridad de los datos. Además, se deben implementar técnicas de manejo de errores robustas para garantizar la estabilidad del sistema en situaciones inesperadas.

Interfaz de Usuario: La solución debe proporcionar una interfaz de usuario intuitiva y fácil de usar para permitir a los operadores del metro y a los usuarios acceder y gestionar la información de manera eficiente. Se deben considerar principios de diseño de interfaces centrados en el usuario para garantizar una experiencia óptima del usuario.

Documentación Exhaustiva: Todos los aspectos del sistema, incluyendo la arquitectura, los algoritmos implementados y las decisiones de diseño, deben estar documentados de manera exhaustiva. Esto facilitará la comprensión del sistema y proporcionará una referencia útil para futuras actualizaciones y mantenimiento.

## Diagrama de clases de la solución planteada


Linea
- Estacion** Estaciones
- string NombreLinea
- int ContadorEstaciones
-int Capacidad
+ linea()
+ linea(Estacion** _estaciones, string NombreLin, int NumEstaciones, int MaxCapacidad)
+ void mostrarLinea() const
+ static linea crearLinea()
+ void agregarEstacion(const Estacion& nuevaEstacion, int indice)
+ void eliminarEstacion(const string& nombreEstacion)
+int getContadorEstaciones()const
+ void setEstaciones(Estacion** _estaciones)
+ Estacion**getEstaciones() const
+void setEstaciones(Estacion** _estaciones)
+ Estacion**getEstaciones() const
+ void seleccionarEstacionTransferencia(const string& nombreEstacion)
+string getNombreLinea() const
+int calcularTiempoEntreEstaciones(const string& nombreEstacionInicio, const string& nombreEstacionFin) const
1
N
Estacion
-int tiempo_anterior
- int tiempo_siguiente
-string nombre
-bool transferencia
+Estacion()
+Estacion(int anterior, int siguiente, string nombre,bool esttransferencia)
+int getTiempoAnterior() const
+ void setTiempoAnterior(int anterior)
+ int getTiempoSiguiente() const
+ void setTiempoSiguiente(int siguiente)
+ string getNombre() const
+ void setNombre(string nombre)
+ bool getTransferencia() const
+ void setTransferencia(bool esttranferencia)

**Algoritmos implementados debidamente intra-documentados:**

El sistema consta de clases que representan las entidades principales: Estación y Línea. Estas clases encapsulan la información relevante sobre las estaciones individuales y las líneas que componen el sistema de metro.

**Constructores y Destructores:**

Constructor de Estación:

Descripción: Inicializa una instancia de la clase Estación con los valores proporcionados para el tiempo anterior, tiempo siguiente, nombre y estado de transferencia.

Parámetros: anterior, siguiente, nombre, esttransferencia.

Destructor de Estación:

Descripción: Libera los recursos asignados a una instancia de Estación cuando ya no se necesita, incluida la memoria para atributos dinámicos.

Constructor de Línea:

Descripción: Crea una nueva instancia de la clase Línea con los valores proporcionados para el arreglo de estaciones, nombre de la línea, número de estaciones y capacidad máxima.

Parámetros: _estaciones, NombreLin, NumEstaciones, MaxCapacidad.

**Getter y Setter:**

Getter de Estación:

Descripción: Permite acceder al valor de un atributo específico de una instancia de Estación.

Parámetros: Ninguno.

Retorno: Valor del atributo solicitado.

Setter de Estación:

Descripción: Modifica el valor de un atributo específico de una instancia de Estación.

Parámetros: Nuevo valor para el atributo a modificar.

Getter de Línea:

Descripción: Similar al getter de Estación, permite acceder al valor de un atributo específico de una instancia de Línea.

Parámetros: Ninguno.

Retorno: Valor del atributo solicitado.

Setter de Línea:

Descripción: Modifica el valor de un atributo específico de una instancia de Línea.

Parámetros: Nuevo valor para el atributo a modificar.

**Métodos de Manipulación de Línea:**

Agregar Estación a Línea:

Descripción: Agrega una nueva estación a la línea en la posición especificada.

Parámetros: nuevaEstacion, indice.

Eliminar Estación de Línea:

Descripción: Elimina una estación de la línea según su nombre.

Parámetros: nombreEstacion.

**Lógica de Transferencia entre Estaciones:**

Seleccionar Estación de Transferencia:

Descripción: Identifica una estación como un punto de transferencia entre líneas.

Parámetros: nombreEstacion.

## Problemas de desarrollo que se afrontó.

Gestión de Memoria Dinámica:

Desafío: El uso de punteros y la asignación dinámica de memoria pueden conducir a problemas de gestión de memoria, como fugas de memoria o acceso a memoria no válida.

Solución: Se requiere una cuidadosa gestión de la memoria, incluida la liberación adecuada de la memoria asignada y la prevención de fugas de memoria mediante la liberación de recursos cuando ya no son necesarios.

Complejidad Algorítmica:

Desafío: Algunas operaciones, como la identificación de estaciones de transferencia entre líneas o la reorganización de estaciones dentro de una línea, pueden requerir algoritmos complejos y propensos a errores.

Solución: Es necesario realizar una planificación cuidadosa y pruebas exhaustivas para garantizar la corrección y eficiencia de los algoritmos implementados. Además, se debe documentar detalladamente la lógica detrás de estos algoritmos para facilitar su comprensión y mantenimiento.

Optimización del Rendimiento:

Desafío: A medida que la red de metro crece en tamaño, las operaciones de gestión de líneas y estaciones pueden volverse más costosas en términos de tiempo de ejecución y recursos del sistema.

Solución: Se deben implementar algoritmos y estructuras de datos eficientes para garantizar un rendimiento óptimo del sistema, incluso en escenarios de carga pesada. Esto puede incluir el uso de técnicas de optimización de algoritmos, como la reducción de la complejidad temporal y el uso de estructuras de datos adecuadas para minimizar el tiempo de búsqueda y manipulación de datos.

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

Escalabilidad del Sistema:

Evolución de la Solución: A medida que la red de metro crece y se expande, la solución debe ser capaz de escalar para manejar un mayor número de líneas, estaciones y pasajeros.

Consideraciones de Implementación: Se deben diseñar los componentes del sistema de manera modular y extensible para facilitar la incorporación de nuevas funcionalidades y adaptarse a cambios en la infraestructura del metro. Esto puede incluir el uso de patrones de diseño como el patrón de fábrica o la inyección de dependencias para desacoplar componentes y permitir una mayor flexibilidad.

Integración de Sistemas Externos:

Evolución de la Solución: Con el tiempo, puede ser necesario integrar el sistema de gestión de líneas y estaciones del metro con otros sistemas externos, como sistemas de información al público o sistemas de gestión de activos.

Consideraciones de Implementación: Se deben establecer interfaces claras y protocolos de comunicación estándar para facilitar la integración con sistemas externos. Además, se deben implementar mecanismos de seguridad para proteger la integridad de los datos y garantizar la integración entre sistemas.

Evolución de la Solución: A lo largo del tiempo, es probable que surjan nuevas necesidades y requisitos para el sistema de gestión del metro, lo que requerirá actualizaciones y mantenimiento continuo.

Consideraciones de Implementación: Se debe establecer un proceso robusto de gestión de cambios que incluya la identificación y priorización de nuevas funcionalidades, la implementación de actualizaciones y parches de seguridad, y la realización de pruebas exhaustivas para garantizar la estabilidad del sistema. Además, se debe mantener una documentación actualizada para facilitar el mantenimiento y la colaboración entre equipos de desarrollo.