UNIVERSIDAD DE ANTIOQUIA

Facultad de Ingeniería

Desafío II

SIMULACIÓN DE UNA RED DE METRO

Michael Gómez Rojas
INFORMATICA II | UNIVERSIDAD DE ANTIOQUIA | 2024

A. Introducción

El Metro es uno de los sistemas de transporte más utilizado en las grandes ciudades, siendo eficaz para trasladar grandes masas de usuarios en poco tiempo. Se requiere que usted diseñe e implemente un simulador de Metro que permita modelar algunas características del funcionamiento de este sistema de transporte.

B. Objetivos

- Desarrollar la capacidad de solución de problemas en los estudiantes, enfrentándolos a problemáticas de la vida cotidiana.

- Verificar si el estudiante adquirió las destrezas y conocimientos fundamentales de la programación orientada a objetos en C++: abstracción, encapsulación, relaciones básicas, diseño de diagrama de clases, funciones amigas, sobrecarga y uso de plantillas.

C. Marco teórico

Un Metro puede concebirse como una red (la red Metro). Una red, como la que se muestra en la Figura 1, es un conjunto de líneas dentro de un área específica. Una línea se identifica por su nombre. A cada línea se le especifica un tipo de transporte: tren o tranvía, así como la secuencia de estaciones que comunica. Una línea no tiene bifurcaciones ni bucles, sin embargo, el tránsito dentro de cualquier línea es bidireccional.

Una estación está identificada por su nombre. Para cada estación, también se conoce el tiempo en segundos que toma llegar desde la estación actual, hasta la estación siguiente o a la estación anterior dentro de su misma línea. Algunas estaciones pueden tener una categoría especial, llamándose estaciones de transferencia. Las estaciones de transferencia representan puntos de intersección entre dos líneas diferentes de la red. Por ejemplo, en el metro de Medellín la estación San Antonio es una estación de transferencia donde confluyen San Antonio A, San Antonio B y San Antonio-Tran.

A efectos de esta simulación de metro, existen elementos de tipo estación, línea y red, con una relación jerárquica clara entre ellos. Tenga presente:

- Una estación puede pertenecer a varias líneas (si es una estación de transferencia). Los nombres de las estaciones de transferencia se conforman concatenando el nombre de la estación con el nombre de la línea donde se encuentra.

- Una línea sólo puede pertenecer a una red.

- Una estación sólo puede estar una vez en una línea.

- Una línea sólo puede estar una vez en una red.

- Si una red tiene más de una línea, estas líneas no pueden estar desconectadas.

Para efectos de este simulador, sólo se construirá una instancia de red Metro.

Figura 1. Red del Metro de Ciudad de México. Tomado de: https://www.metro.cdmx.gob.mx.

El modelo de datos a desarrollar para el simulador debe basarse en el paradigma de POO. Adicional a los constructores, destructores, getters, setters y operaciones de despliegue necesarias; incluya subprogramas que permitan:

A. Agregar una estación a una línea, en los extremos o en posiciones intermedias.

B. Eliminar una estación de una línea. No se pueden eliminar estaciones de transferencia.

C. Saber cuántas líneas tiene una red Metro.

D. Saber cuántas estaciones tiene una línea dada.
E. Saber si una estación dada pertenece a una línea específica.
F. Agregar una línea a la red Metro.
G. Eliminar una línea de la red Metro (sólo puede eliminarse si no posee estaciones de transferencia).
H. Saber cuántas estaciones tiene una red Metro (precaución con las estaciones de transferencia).

Por último, se necesita un subprograma que permita hacer una sencilla simulación del funcionamiento de la red. Dicha simulación ("Cálculo del tiempo de llegada"), toma la hora actual como tiempo de salida de un tren desde una estación de la red, y se debe predecir el tiempo que tardaría ese tren específico en llegar a otra estación dentro de la misma línea. Este subprograma solo realiza esta tarea entre estaciones de la misma línea.

D. Especificaciones

El cliente requiere que usted desarrolle un prototipo de simulador de Red de Metro. El simulador debe permitir representar la estructura de una red metro cualquiera, así como ejecutar las operaciones básicas especificadas en este enunciado a través de un menú.

Se sugiere altamente delimitar el desarrollo a lo estrictamente requerido ya que la dimensión real del problema puede tener una complejidad muy superior al tiempo estipulado para la entrega. En caso de duda sobre los requerimientos, consulte con el cliente.

De acuerdo con lo anterior, entre otras cosas, usted deberá:

1. [10%] Contextualice el problema, analícelo y finalmente diseñe el diagrama de clases correspondiente a su solución. Refleje adecuadamente las relaciones implícitas en la problemática. Recuerde utilizar la notación UML simplificada impartida en las clases teóricas. A pesar de tener una baja ponderación, la entrega del diagrama de clases es obligatoria y sin ella no se dará lugar la sustentación.

2. [10%] Seleccione debidamente los tipos y estructuras de datos que le permitirán implementar las respectivas clases planteadas en 1.

3. [10%] Previo a la implementación, verifique el cumplimiento del requisito de eficiencia especificado en la sección 3 del apartado "Requisitos mínimos" de este documento.

4. [10%] Implementar un subprograma que permita ejecutar la funcionalidad "Cálculo del tiempo de llegada".

5. [60%] Presente la implementación del simulador, cuya interacción se centra en un menú que permita acceder de manera independiente a las funcionalidades correspondientes a cada clase. Considere las prelaciones implícitas en la lógica del problema. Por ejemplo, una estación no se puede crear en el aire, sólo puede crearse si está adscrita a una línea existente en la cual dicha estación no exista ya

E. Desarrollo

1. [10%]

El problema plantea la simulación de una red de metro utilizando programación orientada a objetos (POO). Para representar esta simulación, se diseñó un diagrama de clases utilizando la notación UML simplificada.

El diagrama de clases consta de tres clases principales:

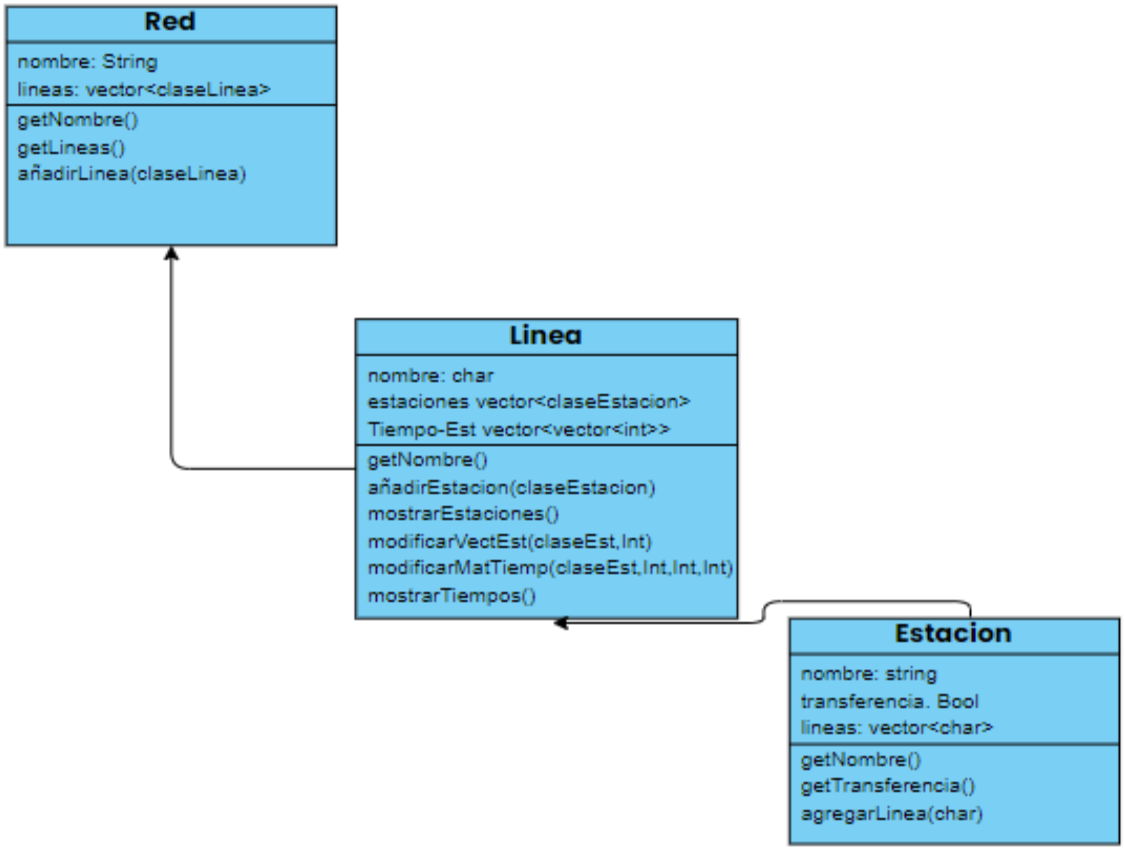

Figura 2. Diagrama de clases.

- **Estación**

Representa una estación en una línea de metro. Tiene atributos como nombre, tiempo de llegada a la siguiente estación y tiempo de llegada desde la estación anterior. Se modela una relación de asociación con la clase Línea para indicar a qué línea pertenece la estación.

1. Atributos:

nombre: Una cadena que representa el nombre de la estación.

transfencia: Una cadena que representa el tipo de la estación (por ejemplo, estación de transferencia).

lineas: Un vector de enteros que representa las líneas de viaje de la red.

- **Línea**

Representa una línea de metro, identificada por un nombre y un tipo (tren o tranvía). Tiene una colección de estaciones y métodos para agregar y eliminar estaciones, así como para calcular el número de estaciones en la línea.

1. Atributos:

nombre: Una cadena que representa el nombre de la línea.

estaciones: Un vector de punteros a objetos de la clase Estacion. Representa todas las estaciones en la línea.

2. Métodos:

agregarEstacion(): Agrega una estación a la línea.

mostrarEstacion

- **Red Metro**

Representa la red de metro en su conjunto. Contiene una colección de líneas y métodos para agregar y eliminar líneas, así como para calcular el número total de líneas y estaciones en la red.

1. Atributos:

lineas: Un vector de punteros a objetos de la clase Linea. Representa todas las líneas de metro en la red.

2. Métodos:

agregarLinea(): Agrega una línea a la red.

	Niquia	Bello	Madera	Acevedo
Niquia	0	50	0	0
Bello	50	0	70	0
Madera	0	70	0	120
Acevedo	0	0	120	0

Figura 3. Matriz de tiempo. Boceto.

2. [10%]

- **Estación**

Para los tiempos de viaje entre estaciones, un vector sería una elección adecuada.

Cada elemento del vector representará el tiempo de viaje desde la estación actual a otra estación en la misma línea. Esto facilitará la gestión de los tiempos de viaje y su acceso durante la simulación.

- **Línea**

Al igual que en el caso de la clase Red, se puede utilizar un vector para almacenar las estaciones en una línea de metro.

- **Red Metro**

La estructura de datos adecuada para almacenar las líneas de metro sería un vector, ya que proporciona acceso aleatorio eficiente y es fácil de manejar.

5. [60%]

```
Bienvenido al simulador de red metro
Digite el nombre de la red: Medellin
¿Desea añadir una linea a esa red? ponga s si es asi o n si no es asi: s
Ingresar nombre de la linea, la linea solo debe de ser una sola letra: A
¿Desea añadir una Estacion a la linea A? ponga s si es asi o n si no es asi: s
Ingresar nombre de la estacion: Bello
En esta linea hay 0 estaciones, indicar en que posicion se ingresara esta estacion: 1
0
Bello
¿Quieres crear otra estacion en la misma linea? Poner s si asi lo desea o n si no es asi: s
Ingresar nombre de la estacion: Caribe
En esta linea hay 1 estaciones, indicar en que posicion se ingresara esta estacion: 2
1
Cuanto tiempo hay entre la estacion Caribe y la estacion Bello
600
0  600
600  0
Bello
Caribe
¿Quieres crear otra estacion en la misma linea? Poner s si asi lo desea o n si no es asi: s
Ingresar nombre de la estacion: Madera
En esta linea hay 2 estaciones, indicar en que posicion se ingresara esta estacion: 2
2
Cuanto tiempo hay entre la estacion Madera y la estacion Bello
200
Cuanto tiempo hay entre la estacion Madera y la estacion Caribe
100
0  200  0
200  0  100
0  100  0
Bello
Madera
Caribe
¿Quieres crear otra estacion en la misma linea? Poner s si asi lo desea o n si no es asi: n
¿Quieres crear otra linea? Poner s si asi lo desea o n si no es asi: s
Ingresar nombre de la linea, la linea solo debe de ser una sola letra: B
```

Figura 4. Menú implementado en simulador.

F. Conclusiones

Al terminar este proyecto sobre la simulación de una red de metro en C++, fue posible aprender en gran medida sobre cómo usar la programación para representar cosas reales, como estaciones y líneas de metro. Durante el proyecto, se encontraron con desafíos interesantes, como organizar las clases y hacer que el simulador funcione correctamente. Pero a medida que se avanzaba, también se iba mejorando la habilidad para resolver problemas y pensar de forma lógica.

Trabajar en este proyecto también permite practicar programación en C++, usando cosas como vectores y algoritmos para calcular tiempos de llegada entre estaciones. Fue una oportunidad emocionante para aplicar lo que se había aprendido en clase a un proyecto real y desafiante. Al final, no solo se mejoraron sus habilidades técnicas, sino que también se ganó una mayor comprensión de cómo funciona la programación orientada a objetos en C++.

Se logró diseñar un sistema completo de acuerdo a lo que se solicitó, a excepción del tiempo total de viaje.

G. Bibliografía

1. Schildt, H. (2017). "C++: The Complete Reference." McGraw-Hill Education.
2. Stroustrup, B. (2013). "Programming: Principles and Practice Using C++." Addison-Wesley.
3. Fu, X., & Liu, J. (2015). "Discrete-Event Simulation: A First Course." Chapman and Hall/CRC.