Universidad de Antioquia.


# Informe del parcial II.

Giraldo Úsuga Mauricio.
Parra Osorio Rafael Ignacio.

Asignatura: Informática II.
Aníbal José Guerra Soler.
Augusto Enrique Salazar Jiménez.

Medellín.
2024.

## Índice:



## Introducción:

En el siguiente informe se pretende plasmar el desarrollo del desafío II de la asignatura de informática II, el cual abordará los temas de análisis y evolución, así como los elementos implementados para llevarlo a cabo.

### a) Análisis del problema y consideraciones:

La construcción de una red de metro implica que siempre exista interconectividad, por lo cual la segunda línea obligadamente debe intersectarse con la primera, luego a partir de la creación de la tercera línea se le dará al usuario las opciones de poder hacer intersección con cualquiera de las líneas existentes que desee sólo una vez, pues no pueden haber bucles según el desafío.
Una manera conveniente para construir la red metro sería la siguiente:
Primero se construyen las líneas con sus respectivas intersecciones (estaciones de transferencia), pues junto con las líneas son el esqueleto de la red al ser inamovibles, luego, en base a las estaciones de transferencia se ponen las estaciones sencillas en las líneas deseadas, ahora con la estructura lista se configuran los tiempos entre estaciones.
Luego del anterior proceso, se puede acceder a todas las opciones del menú, como quitar estaciones, consultar estaciones o crear líneas, entre otras; además se podrán agregar líneas que intersecten con cualquiera de las estaciones creadas previamente, sean de transferencia o sencillas, si se intersecta la línea nueva con una estación sencilla, esta pasará a ser una estación de transferencia y no se podrá eliminar luego.
Las estaciones de transferencia podrán ser la intersección entre dos o más líneas, cunado intersectan en ellas más de dos líneas, serían estaciones centrales.
Ahora, modelando el problema en clases, se podría decir que habría dos clases con las cuales representar dicha realidad, por ejemplo, una clase para las estaciones donde sus atributos serían el nombre y el tiempo de recorrido hasta las líneas adyacentes, la otra clase sería una clase que represente a la red de metro que en

esencia sería el conjunto total de líneas y sus atributos serían valores asociados a la ubicación de sus componentes como filas y columnas.

## b) Diagrama de clases de la solución planteada:

| EstacionMatriz |
| --- |
| (-)Estacion*** datos<br>(-)int filas<br>(-)int columnas |
| (+)EstacionMatriz(int filas, int columnas);<br>(+) ~EstacionMatriz();<br>(+) int cantidadLineas() const;<br>(+)Estacion* getEstacion(int fila, int columna) const;<br>(+)void setEstacion(int fila, int columna, Estacion* estacion);<br>(+)void redimensionar(int nuevaFilas, int columnas);<br>(+)void borrarEstacion(int fila, int columna);<br>(+)int obtenerFilas() const { return filas; }<br>(+)void mostrarMatriz() const;<br>(+)int obtenerColumnas() const { return columnas; }<br>(+)void copiarElemento(int origen, int destino, int<br>(+)indiceElemento);<br>(+)int cantidadColumnasEnFila(int fila) const;<br>(+)bool estacionPerteneceALinea(const string&<br>(+)nombreEstacion, int linea) const;<br>(+)int cantidadEstacionesSinTransferencia() const; |

| Estacion |
| --- |
| (-)string nombre<br>(-)int tiempo |
| (+)Estacion(const string&)<br>(+) ~Estacion()<br>(+)string getNombre() const<br>(+) void setNombre(const string&<br>(+)nuevoNombre)<br>(+)void getEstacion() const<br>(+)bool operator==(const Estacion&<br>(+)otraEstacion) const<br>(+)Estacion& operator=(const Estacion&<br>(+)otraEstacion) |

1

N

## c) Algoritmos implementados:

EstacionMatriz inicializarmatriz(int n) {}

EstacionMatriz inicializarmatriz(int n)

void crearestacion(EstacionMatriz& sistemametro, int fila, int columna)

## d) Problemas de desarrollo afrontados:

Uno de los problemas que se ha presentado, es el concepto de «Estación central» que es cuando una estación de transferencia es intersectada por más de dos líneas (tres líneas o más), ya que al principio daba la impresión de entrar en conflicto con las instrucciones del desafío, pues en este se especifica que no están permitidas las bifurcaciones de las líneas, pero, al ser una estación central, las ramas que salen de esta no son bifurcaciones de una línea, sino líneas diferentes. Esto es un problema a la hora de agregarlo a la lista de las estaciones, ya que se tendría que cambiar la nomenclatura de las estaciones de transferencia existentes y su tipo de funcionamiento, porque al pasar a ser estaciones de transferencia no podrían ser eliminadas.
Otra dificultad ha sido la función de contar estaciones totales, puesto que las estaciones de transferencia se repiten en los arreglos, ya que son intersecciones entre líneas.
Definir las estaciones de transferencia ha sido otra dificultad, pues hasta el momento se concatena el nombre de la línea donde va a llegar más no con la línea de origen.
Pero el mayor de los problemas que hemos afrontado es la implementación del tiempo entre estaciones, ya que hay dificultades para implementarlo, luego se necesita hallar la manera de cómo hacer coincidir los arreglos de los tramos de tiempo con los arreglos de las estaciones cuando estas son modificadas.

## e) Evolución de la solución y consideraciones:

Al principio se pensó en modelar el problema con dos clases llamadas RedMetro y Linea, donde Linea iba a ser un arreglo de strings y RedMetro iba a ser una matriz de strings de tres dimensiones, ya que se cada string es un arreglo de caracteres (sería un arreglo de arreglos de arreglos de caracteres), en cuanto a las estaciones, en un principio se pensó representarlas tan sólo como unos arreglos dinámicos que se podrían modificar, al igual que el tiempo entre estaciones, ambos arreglos serían de dos dimensiones (un arreglo de arreglos) en donde las filas serían las líneas y las columnas las estaciones, en cuanto al arreglo 2D del tiempo, cada fila de este tendría un tamaño de n – 1 en comparación con cada fila del arreglo de estaciones, pues los tramos de tiempo en comparación con las estaciones totales de una línea siguen el patrón antes mencionado.
Más adelante se replanteó el modelado del problema y se optó por representarlo en dos clases llamadas EstacionMatriz y Estacion, donde la primera representa la red de metro que depende de la segunda y tienen una relación de 1 a muchos, los atributos de EstacionMatriz son una matriz de tres dimensiones (porque en cada

fila y en cada columna se guarda un objeto de una clase que es un puntero a string) y dos int que son filas y columnas, mientras que la clase estación tiene en sus atributos un puntero a string para el nombre y un int para el tiempo, ambas clases con atributos privados.
La dificultad de concatenar en el origen para definir las estaciones de transferencia se ha solucionado arreglando el método copiarElemento, haciendo que también concatene en la estación de origen.

## **Conclusión:**

Una lección importante que este desafío nos ha dejado, es que el análisis de un problema de la realidad puede dar lugar a una gran variedad de modelados y abstracciones para ser interpretada por un algoritmo en un lenguaje de programación, en donde se requiere comparar cada uno de los candidatos de modelado para poder escoger el más acertado, porque de hacer una mala elección, es decir, un modelado mediocre, se podría comprometer seriamente la practicidad en la elaboración del código e incluso su eficiencia, ahora no es solo pensar antes de comenzar a construir la primera línea de código, sino que hay que pensar, analizar, armar un abanico de posibilidades y luego hacer una muy buena elección.