Estructuras de Datos y de la Información

2ª Entrega. Curso 2021/22

Red Social de Viajes [RSV]

Por:
Profesores de EDI

Índice general

1. Introducción

Este documento describe los requerimientos y funcionalidad básica para el desarrollo de la segunda entrega de programación de la asignatura *Estructuras de Datos y de la Información* del curso 2021-22.

En esta entrega vamos a implementar una aplicación para gestionar una Red Social de Viajes compartidos (RSV).

Motivación:
Los viajes compartidos en vehículos privados han captado gran interés por parte del público en los últimos años. Compartir viaje con personas que se dirigen hacia los mismos destinos genera beneficos económicos y ecológicos. El principal beneficio a nivel individual es el ahorro económico, derivado de compartir los gastos del viaje.

El objetivo será desarrollar una aplicación para agilizar la gestión de un servicio para compartir vehículo (similar a Blablacar).

Para el desarrollo de la aplicación conforme a los requisitos de la asignatura se seguirá una metodología de desarrollo basada en el paradigma de *Programación Orientada a Objetos* y se utilizará el lenguaje de programamción *C++*.

Este documento se organiza de la siguiente forma. En la sección 2 se discuten de forma general los pasos que debemos seguir para el desarrollo de la aplicación y se ofrecen detalles de implementación, incluido los contenedores de datos que usaremos en esta parte, la *Lista con Punto de Interés*, las *Pilas y Colas* . En la sección 3 se enumeran los algoritmos básicos que debemos implementar y en la sección **??** se establece la documentación interna de las clases. Finalmente, la sección 5 establece los criterios de evaluación de la calidad del desarrollo y su calificación, así como la fecha de entrega.

2. Metodología e Implementación

En esta sección se describe la funcionalidad que implementará nuestra aplicación, y se establecen una serie de pasos generales que pueden servir de guía para el desarrollo de la misma.

En el desarrollo de esta segunda entrega, se partirá de un proyecto, que podéis descargar del aula virtual *RSV.zip*. El proyecto contiene los ficheros de texto, que se deben cargar en la aplicación, las plantillas de las estructuras de datos y el esqueleto de la clase principal (RSV) en la que se deben incluir los algoritmos a implementar.

2.1. Ficheros de Datos

Los datos con los que trabajaremos se proporcionan en tres ficheros de texto.

Estos ficheros de describen en el Cuadro 1. Los campos de cada fichero son autoexplicativos, es labor del programador extraer el tipo de cada dato para representarlo adecuadamente.

Nombre	**Descripción**
`conductores.csv`	Fichero que contiene los *conductores* disponibles en la RSV
`valoraciones.csv`	Fichero que contiene *valoraciones/opiniones* a los conductores .
`viajes.csv`	Fichero que contiene los posibles *viajes* que se ofrecen en la RSV.

Cuadro 1: Ficheros de datos para la aplicación.

La información de los ficheros se cargará en las estructuras de datos de la aplicación al **inicio de la ejecución** (antes de mostrar el menú por pantalla).

2.2. Implementación de las clases *base*

Las clases básicas a implementar en la aplicación serán las que representen un Conductor, un Viaje y una Opinión. Deben contener los atributos (privados) necesarios para la funcionalidas de la aplicación y las operaciones de accceso a los atributos y al menos una operación para mostrar la información en pantalla. A estas operaciones básicas se podrán añadir otras si es necesario. Información detalla de cada clase:

Conductor: Para cada conductor se debe guadar la siguiente información: Identificador del conductor (string), nombre, edad, año en el que se dio de alta en la aplicación y marca y modelo del coche. Además debe tener un atributo donde se puedan almacenar las valoraciones/opiniones de los usuarios, donde la mas reciente es la primera que se muestra.

Opinión: Para cada opinión se debe guardar la siguiente información: la valoración del servicio prestado, será un entero entre 1 y 10, donde el 1 representa un servicio pésimo y el 10 excelente, y la opinión del servicio prestado (string).

Viaje: Para cada viaje se debe guardar la siguiente información: Ciudad de origen (string), ciudad de destino (string), hora de salida (int por simplificar) y el identificador del conductor.

Por simplificar el problema, no sera tenida en cuenta la fecha de los viajes ni el número de pasajeros.

2.3. Contenedores de datos

Las estructuras de datos básicas que vamos a utilizar en nuestra aplicación son la *Lista con Punto de Interés*, la *Pila* y la *Cola*. La implementación de dichos contenedores será proporcionada por los profesores de la asignatura, y se deben usar sus interfaces sin modificaciones.

La E.D. `LPI` se utilizará para almacenar los datos de los conductores y de los viajes en la aplicación, la E.D. `Pila` se utilizará para almacenar las valoraciones/opiniones sobre cada conductor. Será necesario utilizarlas correctamente siguiendo la sintaxis basada en *templates* de C++. Además, las *Precondiciones* y *Postcondiciones* para cada una de las operaciones se deben respetar por parte del programador[1].

Los criterios de ordenación de los datos en las estructuras de datos de la aplicación debe quedar claramente definido. Los *conductores* se ordenarán alfabéticamente por el identificador y los *viajes* alfabéticamente por el nombre de la ciudad de origen y por la hora de salida ascendentemente.

2.4. Implementación de la clase principal

Tendremos una clase principal que represente la RSV, y que contendrá como atributos dos puntores a las E.D. que contienen a todos los conductores y viajes de la aplicación.

Atributos:

1. LPI<Viaje *>*lv.
2. LPI<Conductor *>*lc.

Recordad que el constructor debe ser el encargado de cargar la información desde los ficheros de texto y el destructor debe liberar todos los recursos asociados a los atributos.

[1]Se recuerda que las pre/post-condiciones son expresiones booleanas, y no texto descriptivo, y que una vez establecidas deben ser respetadas, es decir, el programador que utiliza la interfaz debe cumplir las precondiciones, y el programador que implementa la interfaz debe asegurar las postcondiciones.

3. Algoritmos

Los algoritmos a implementar en nuestra aplicación se deben mostrar en un menú inicial. Antes de mostrar el menú, se deben cargar los datos de la aplicación en las estructuras de datos contenedoras desde los ficheros de texto que contienen los conductores y los viajes.

Después de mostrará el menú con las siguientes opciones:

1. **Insertar conductor**. La operación recibirá los datos de un nuevo conductor (identificador, nombre, edad, coche y año de alta). Debe crear el conductor e insertarlo en orden alfabético en la lista por el identificador. En el caso de que existiera un conductor con el mismo identificador no se insertará.
2. **Valorar a un conductor**. La operación recibirá el identificador del conductor a valorar, la valoración y la opinion del servicio prestado. La operación debe buscar el conductor y añadirle la valoración/opinion.
3. **Insertar viaje**. La operación recibirá los datos de un nuevo viaje (ciudad de origen, destino, hora de salida e identificador del conductor). Debe crear el viaje e insertarlo en orden en la lista. El orden lo determina la ciudad de origen y la hora de salida.
4. **Mostrar conductores**. Muestra por pantalla todos los conductores ordenados alfabéticamente por el identificador del conductor. Por cada conductor se mostrará su **valoración media**.
5. **Mostrar viajes**. Muestra por pantalla todos los viajes ordenados por ciudad de origen y hora de salida.
6. **Mostrar todo**. Muestra por pantalla todos los viajes con la información del conductor.
7. **Buscar conductor**. Debe buscar un determinado conductor por su identificador, la operación devolverá toda la información del conductor para posteriormente mostrarla por pantalla incluyendo todas sus valoraciones/opiniones, desde la mas reciente a la mas antigua.
8. **Eliminar conductor**. Debe eliminar un determinado conductor por su identificador. Si en conductor tiene viajes asociados también se deben eliminar, al igual que todas sus valoraciones/opiniones.
9. **Cola viajes**. Devuelve una cola con todos los viajes que tengan un origen y destino iguales a los pasados como parámetros de entrada al método. La cola devuelta se debe mostrar.

Todos los algoritmos deben ser implementados sobre las estructuras de datos en memoria, es decir, no están permitidas las implementaciones de los algoritmos directamente sobre los ficheros de datos.

La interfaz de las operaciones debe ser diseñada y documentada correctamente conforme a la descripción.

4. Documentación

Todos los algoritmos y métodos implementados en la RSV deben estar correctamente documentados. La documentación incluirá, al menos, por cada método las Precondiciones y Poscondiciones, complejidad (utilizando notación Big-O) y descripción de la funcionalidad del método. Además, cada fichero de código contendrá en la parte superior, el nombre del autor y correo electrónico, de de la última modificación y una breve descripción del contenido.

5. Criterios de evaluación y fecha de entrega

La evaluación de la implementación tomará en cuenta los criterios generales que se especifican a continuación:

1. Uso de punteros en la implementación, tanto en variables como en estructuras de datos, evitando así los problemas derivados de mover datos en memoria (*eficiencia*) y mantener distintas copias (*consistencia*).
2. Organización correcta de los datos en memoria de forma que la implementación de los algoritmos cumpla con los requisitos de eficiencia y uso de memoria.
3. Corrección en el resultado de la ejecución de los algoritmos.
4. Utilización correcta de memoria, en cuanto a utilización de la necesaria y reserva/liberación de forma correcta.
5. El lenguaje de programación será C++. No está permitido utilizar la STL[2] en el desarrollo, ni cualquier otra biblioteca similar.
6. Uso correcto y eficiente de parámetros en la invocación de operaciones.
7. Uso de constructores y destructores para las clases de forma coherente y adecuada.

[2]Standard Template Library

8. Siempre que sea posible, no se duplicarán objetos en la aplicación, y se evitarán copias y duplicados de los mismos.

Además, las siguientes normas se deben seguir en la realización del proyecto:

- El proyecto se debe realizar de forma **individual**.
- La entrega del proyecto consistirá en un fichero `zip` cuyo nombre seguirá el formato `ApellidosNombreAlumno.zip`.
- Cualquier intento de copia, detección de entrega de código no propio o desarrollado por terceros, será motivo de suspenso (0) en la asignatura (para todos los alumnos implicados), y puesta en conocimiento de la autoridad del centro.

La fecha límite de entrega del proyecto es el día **3 de abril a las 23:55h** a través de la correspondiente tarea abierta en el campus virtual de la asignatura. No se tendrán en cuenta entregas realizadas por cualquier otro medio que no sea el establecido, ni con posterioridad a esa fecha.