Programación Avanzada 2022

Laboratorio 1

Consideraciones generales:

- La entrega podrá realizarse hasta la fecha indicada en el aula virtual de Programación Avanzada dentro del Campus.
- Las entregas deberán realizarse de acuerdo a las plantillas disponibles en el Campus.
- Las entregas serán realizadas únicamente vía web y se deberán subir usando el Campus del curso. Sólo un miembro del grupo deberá entregar un único archivo que contenga la entrega, el archivo deberá llamarse <número de grupo>_lab1.zip (o tar.gz) que contenga:
  - El código fuente de la aplicación.
  - Un archivo makefile, que permita compilar y ejecutar el código, independiente de cualquier entorno de desarrollo integrado (IDE).
- Las entregas que no cumplan estos requerimientos no serán consideradas.
- El código junto con el makefile se probarán en alguna máquina con Linux en una defensa obligatoria y eliminatoria con todos los integrantes del grupo.
- No se aceptarán entregas fuera del plazo establecido y el hecho de no realizar una entrega implica la insuficiencia del laboratorio completo.

Con este laboratorio se espera que el estudiante adquiera competencias en la implementación de operaciones básicas, el uso básico del lenguaje C++ (que se usará en el laboratorio) y el entorno de programación en linux, así como reafirmar conceptos presentados en el curso. También se espera que el estudiante consulte el material disponible en el Campus del curso y que recurra a Internet con espíritu crítico, identificando y corroborando fuentes confiables de información.

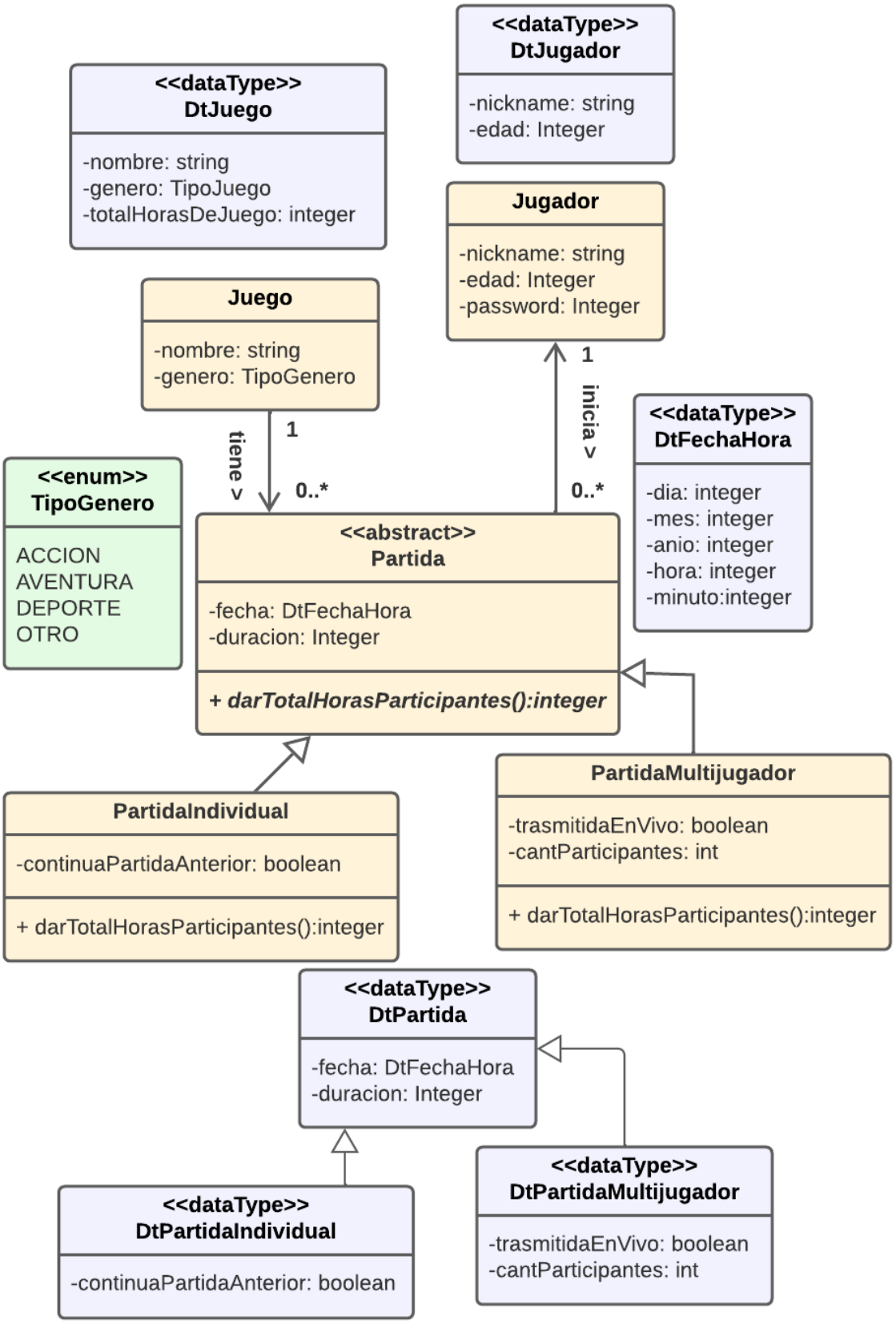
Problema

Se desea implementar un prototipo de sistema para aficionados a los videojuegos, el cual debe contar con con un catálogo de videojuegos de diversos géneros, disponibles exclusivamente para los jugadores registrados en el sistema.

De cada videojuego se registra su nombre, el cual lo identifica, y su género. Los jugadores se registran ingresando un nickname (seudónimo que lo identifica), edad y contraseña, y pueden iniciar tanto partidas individuales como multijugador. Estas últimas pueden ser transmitidas en vivo y se conoce la cantidad de participantes.

El sistema almacena las partidas jugadas registrando la fecha en la que se realizan y su duración (en horas). Para cada partida individual interesa saber además si la misma fue una partida nueva o una continuación de una anterior. Por último, es de interés contar con estadísticas que permitan conocer cuáles son los videojuegos preferidos por los usuarios. Para esto, el sistema permite conocer la cantidad total de horas dedicadas por los jugadores a determinado videojuego.

En base a la descripción anterior se ha diseñado la estructura de clases que muestra la siguiente figura.



Se pide:

Se pide implementar en C++:

1. Todas las clases (incluyendo sus atributos, pseudoatributos, constructores y destructores, y los getters y setters necesarios para las operaciones que se indican más adelante), enumerados y data types que aparecen en el diagrama. Para las fechas en caso de recibir dd > 31 o dd < 1 o mm > 12 o mm < 1 o aaaa < 1900, se debe levantar la excepción std::invalid_argument. No se deben hacer más que estos controles en la fecha (ej. la fecha 31/2/2000 es válida para esta realidad).
2. Las siguientes operaciones/funcionalidades (no se puede cambiar la firma de las mismas):

void agregarJugador(string nickname, int edad, string password)	
Que?	Registra un nuevo jugador en el sistema.
Como?	El usuario ingresa el nickname, la edad y la password y se da de alta el jugador en caso de que no exista en el sistema, si ya existe un jugador registrado con el mismo nickname, se lanza una excepción de tipo std::invalid_argument.

void agregarVideojuego(string nombre, TipoGenero genero)	
Que?	Registra un nuevo videojuego en el sistema.
Como?	El usuario ingresa el nombre y el género del videojuego. Si ya existe un videojuego registrado con el mismo nombre, se lanza una excepción de tipo std::invalid_argument.

DtJugador obtenerJugadores(int& cantJugadores)**	
Que?	Lista los jugadores registrados en el sistema.
Como?	Devuelve un arreglo con información sobre los jugadores registrados en el sistema. El parámetro cantJugadores es un parámetro de salida donde se devuelve la cantidad de jugadores devueltos por la operación (corresponde a la cantidad de instancias de DtJugador retornadas).

DtVideojuego obtenerVideojuegos(int& cantVideojuegos)**	
Que?	Lista los videojuegos registrados en el sistema.
Como?	Devuelve un arreglo con información sobre los videojuegos registrados en el sistema. El parámetro cantVideojuegos es un parámetro de salida donde se devuelve la cantidad de jugadores devueltos por la operación (corresponde a la cantidad de instancias de DtVideojuego retornadas). El atributo totalHorasDeJuego corresponde a la suma de horas jugadas por todos los jugadores del videojuego. Esto debe calcularse sumando las horas devueltas por las invocaciones a darTotalHorasParticipantes() sobre cada partida del juego. Si la

	partida es individual, la cantidad de horas devuelta por dicha operación es igual a su duración, mientras que si es multijugador devuelve su duración multiplicada por la cantidad de participantes de la partida.

DtPartida obtenerPartidas(string videojuego, int& cantPartidas)**	
Que?	Lista las partidas de un videojuego del sistema.
Como?	Devuelve un arreglo con información de las partidas del videojuego identificado por videojuego. El parámetro cantPartidas es un parámetro de salida donde se devuelve la cantidad de partidas devueltas por la operación (corresponde a la cantidad de instancias de DtPartida retornadas). Entre los datos devueltos para ambos tipos de partida se encuentra la duración. Además, entre los específicos de cada partida individual se encuentra si la misma es o no continuación de una partida anterior, mientras que para cada partida multijugador se indica si la misma es transmitida en vivo y la cantidad total de jugadores que se unen a la misma. Si no existe un videojuego registrado con el nombre enviado, se lanza una excepción de tipo std::invalid_argument.

void iniciarPartida(string nickname, string videojuego, DtPartida* datos)	
Que?	Registra una partida
Como?	Registra una partida individual o multijugador del videojuego identificado por videojuego, iniciada por el jugador identificado por nickname. El parámetro de entrada datos contiene la información completa de la partida. Entre los datos comunes a ambos tipos de partida se encuentra la duración. Además, si datos es una instancia de DtPartidaIndividual contiene si la partida es o no una continuación de una partida anterior, mientras que si es un instancia de DtPartidaMultijugador, indica si la partida es transmitida en vivo y la cantidad total de jugadores que se unen a la misma. La partida se da de alta con la fecha del sistema al momento del registro. Si no existe un jugador o videojuego registrado con el nickname y nombre enviados, se lanza una excepción de tipo std::invalid_argument.

Notas:

- A los efectos de este laboratorio, considere que el sistema maneja un conjunto acotado de jugadores, videojuegos, y partidas por videojuego, donde la cota se define por las constantes MAX_JUGADORES, MAX_VIDEOJUEGOS y MAX_PARTIDAS, respectivamente.
- Puede implementar operaciones auxiliares en las clases dadas en el diagrama si considera que le facilitan para la resolución de las operaciones pedidas.
- Se puede utilizar el tipo std::string para implementar los atributos de tipo string.

- En este laboratorio no se pueden utilizar estructuras de datos de la biblioteca STL, tales como vector, set, map, etc.
- Se sugiere **[OPCIONAL]** crear una clase auxiliar llamada Sistema que implemente las operaciones pedidas y almacene el conjunto de jugadores y videojuegos.

3. Implementar un menú sencillo e interactivo con el usuario para probar las funcionalidades requeridas en los puntos anteriores. Al ejecutar el programa debe primero pedir el ingreso de un número especificando la acción a realizar, y luego pedir los datos necesarios para cada operación.

4. **[OPCIONAL]** Sobrecargar el operador de inserción de flujo (ej. <<) en un objeto de tipo std::ostream. Este operador debe "imprimir" todos los datos de las distintos datatypes de DtPartida (DtPartidaIndividual, DtPartidaMultijugador), DtVideoJuego y DtJugador.