Programación Avanzada 2021

Laboratorio 1

Consideraciones generales:

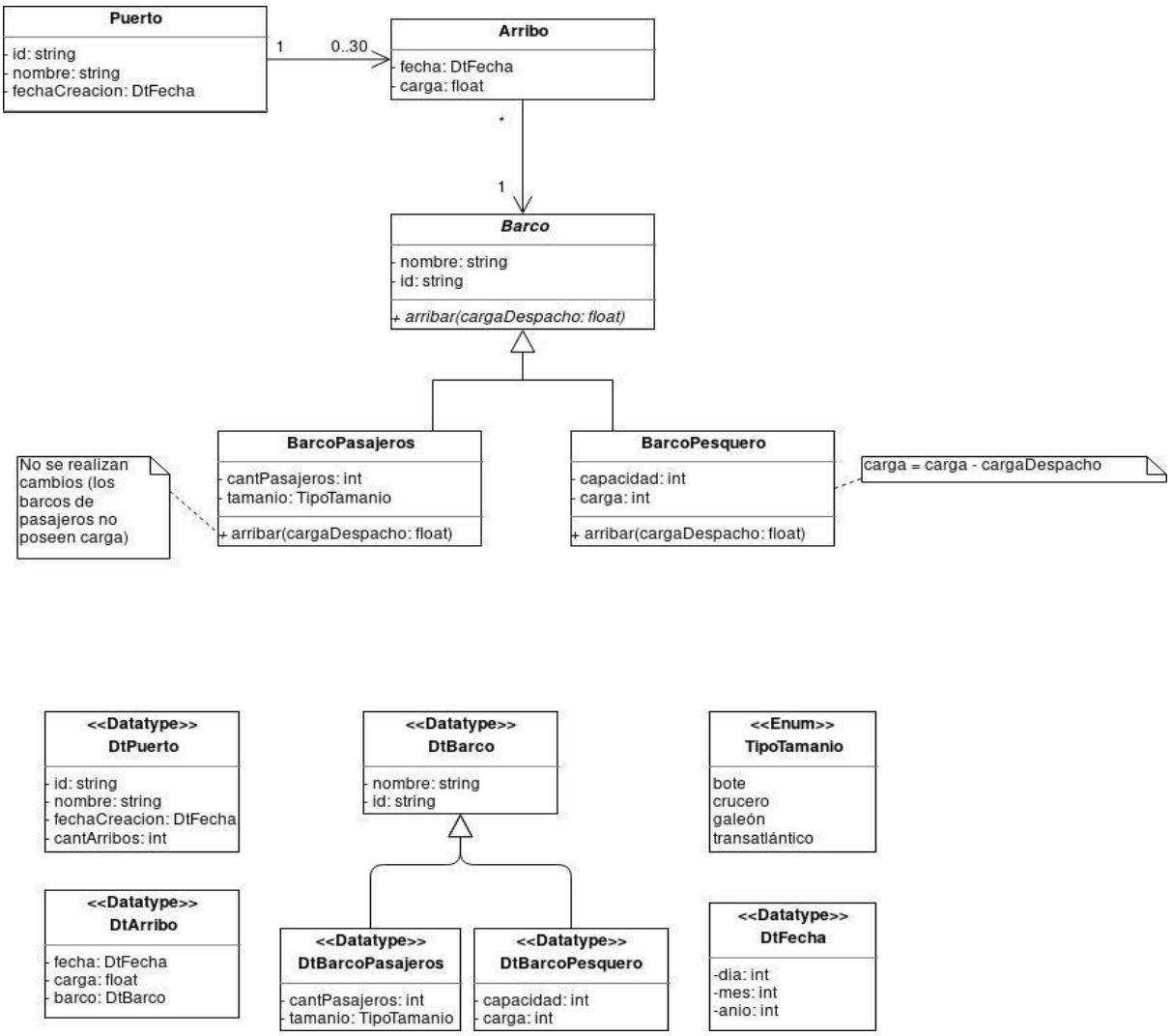
- La entrega podrá realizarse hasta la fecha indicada en el aula virtual de Programación Avanzada dentro del Campus**.**
- Las entregas deberán realizarse de acuerdo a las plantillas disponibles en el Campus.
- Las entregas serán realizadas **únicamente** vía web se deberán subir usando el Campus del 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 los dos ejercicios.
  - Un archivo makefile, que permita compilar y ejecutar el código, independiente de cualquier entorno de desarrollo integrado (IDE).
  - Un archivo denominado resp_lab1.txt, con las respuestas a las preguntas planteadas en el ejercicio 2.
- 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 pequeño sistema para manejar la información de los puertos y los barcos que allí arriban.
De los puertos interesa conocer su identificador (id), su nombre (por ej. Mar del Plata), fecha de creación y los arribos que haya tenido. Para simplificar, se asumirá que en cada puerto no hay más de 30 arribos. De dichos arribos interesa saber el barco que arriba, la fecha y la carga despachada (que se explicará más adelante). Los barcos pueden ser de dos tipos: de pasajeros o pesqueros. De los barcos interesa saber su identificador (id) y su nombre (por ej. El Cutty Sark). En particular, de los barcos de pasajeros interesa además la cantidad de pasajeros y su tamaño mientras que de los barcos pesqueros interesa la capacidad total y su carga.
La carga despachada en los arribos de barcos pesqueros varía, pudiendo ser positiva (en el caso que el barco deje carga en el puerto), negativa (en el caso que retire carga) o cero (si no cambia la carga). La carga despachada en los arribos de barcos de pasajeros siempre es cero.

El siguiente diagrama es una representación de la realidad descrita.



Ejercicio 1

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 (**no se puede cambiar la firma de las mismas**):

    a. `void agregarPuerto(string id, string nombre, DtFecha fechaCreacion)`

    Agrega un nuevo puerto en el sistema. En caso de que ya exista un puerto con el mismo identificador, levanta la excepción `std::invalid_argument`.

b. `void agregarBarco(DtBarco& barco)`

   Agrega un nuevo barco en el sistema. En el caso de que exista un barco con el mismo identificador, levanta una excepción *`std::invalid_argument`.*

c. `DtPuerto** listarPuertos(int& cantPuertos)`

   Devuelve un arreglo con información de los puertos registrados en el sistema. El parámetro cantPuertos es un parámetro de salida donde se devuelve la cantidad de puertos devueltos por la operación (corresponde a la cantidad de instancias de DtPuerto retornadas).

d. `void agregarArribo(string idPuerto, string idBarco, DtFecha fecha, float cargaDespacho)`

   Agrega un arribo al puerto con identificador idPuerto correspondiente al barco con identificador IdBarco en la fecha actual y cargaDespacho como valor de la carga que se despacha. Si el valor de carga que se despacha es positivo (el barco descarga en el puerto), se debe disminuir el valor de carga del barco en esa cantidad. Si el valor de carga que se despacha es negativo o 0, se debe aumentar el valor de carga del barco en esa cantidad.Cualquiera de las siguientes situaciones es motivo para levantar una excepción *`std::invalid_argument`:*
   - No existe un barco con identificador `idBarco`
   - No existe un puerto con identificador `idPuerto`
   - Se aumenta la carga de un barco más allá de la capacidad total que soporta
   - El barco no tiene suficiente carga para realizar el arribo
   - El barco es de pasajeros y la carga que se despacha es distinta de cero.

En caso de que se levante la excepción debe asegurarse que el estado del sistema sea el mismo que aquel previo a la operación.

e. `DtArribo** obtenerInfoArribosEnPuerto(string idPuerto, int& cantArribos)`

   Devuelve un arreglo con información de los arribos registrados en el sistema para el puerto identificado por `idPuerto`. El parámetro cantArribos es un parámetro de salida donde se devuelve la cantidad de arribos devueltos por la operación (corresponde a la cantidad de instancias de DtArribo retornadas). En caso que no exista un puerto con `idPuerto` se levanta una excepción `std::invalid_argumen`*`t`.*

f. `void eliminarArribos(string idPuerto, DtFecha fecha)`

   Elimina los arribos en el puerto identificado por `idPuerto` de la fecha ingresada fecha. En caso que no exista un puerto con `idPuerto` se levanta una excepción `std::invalid_argumen`*`t`.*

g. `DtBarco** listarBarcos(int& cantBarcos)`

Devuelve un arreglo con información de los barcos registrados en el sistema. El parámetro cantBarcos es un parámetro de salida donde se devuelve la cantidad de barcos devueltos por la operación (corresponde a la cantidad de instancias de DtBarco retornadas).

Notas:

- A los efectos de este laboratorio, considere que el sistema manejará un conjunto de puertos de a lo sumo una cantidad `MAX_PUERTOS` (constante) así como un conjunto de barcos de a lo sumo `MAX_BARCOS` (también constante).
- Puede implementar operaciones 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.

3. Implementar un menú sencillo e interactivo con el usuario para probar las funcionalidades requeridas en el punto 2 y 3. 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.

```
Ejemplo:
      Bienvenido. Elija la opción.
         1) Agregar puerto
         2) Agregar barco
         3)
         4) Listar puertos
         5) ….
         6) Salir
      Opción:
```

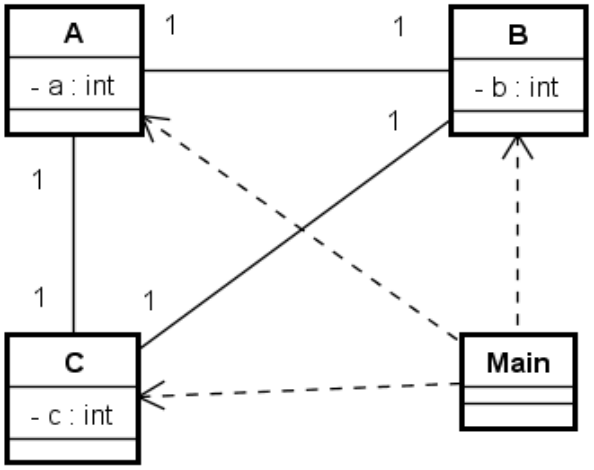
Se deben contemplar las excepciones que se producen en las operaciones. La única acción que puede terminar el programa es usar la opción “Salir”

4. Sobrecargar el operador de inserción de flujo (ej. <<) en un objeto de tipo `std::ostream`. Este operador debe “imprimir” las distintas clases de DtBarco (DtBarcoPesquero, DtBarcoPasajeros) con el siguiente formato:

```
- Id barco: id
- Nombre: nombre
- Tipo de barco: nombre del tipo de barco
      (en el caso de barco de pasajeros)
      Cantidad de pasajeros: cantPasajeros
      Tamaño: tamanio
      (en el caso de barco pesquero)
      Capacidad: capacidad
      Carga: carga
```

Ejercicio 2

En este ejercicio se busca que se familiarice con la problemática de dependencias circulares en C++. Para esto se debe implementar y compilar la siguiente estructura dada por el diagrama a continuación.



Se pide:

1. Implementar, compilar y ejecutar en C++ el diagrama dado. Defina un atributo y constructores en las clases A, B y C así como operaciones que impriman un mensaje con el nombre de la clase y el valor del atributo. Defina un main que cree objetos de esas clases e invoque a dichas operaciones.
2. Responder las siguientes preguntas:
    a. ¿Cuáles son las dependencias circulares que fueron necesarias solucionar para que el programa compile?
    b. ¿Qué es forward declaration?