Homework 3 – Serialización y Programación Genérica

I102 – Paradigmas de Programación
Fecha límite de entrega: 31/05/2025 – 22hs

La entrega se realizará mediante un archivo de texto que contenga el enlace a un repositorio privado de GitHub con la solución de cada ejercicio, organizados en carpetas separadas. Además, el archivo debe incluir un texto que indique cómo compilar el código de cada ejercicio. Cada carpeta de los ejercicios debe contener únicamente archivos sin compilar, como .h, .hpp, .cpp, entre otros (también se puede incluir un archivo .gitignore).

1. Programe un sistema de clases que almacene dos mediciones de vuelo: posición (x, y, z) y presión (presión estática y presión dinámica). Cada medición debe incluir el tiempo en que se realizó.
Este sistema debe de ser capaz de guardar y recuperar la información desde un único archivo mediante la implementación manual de serialización y deserialización de los objetos involucrados. **Importante**: usted debe implementar la lógica de serialización y deserialización sin utilizar librerías externas.
Su diseño debe seguir el indicado en la Fig. 1.

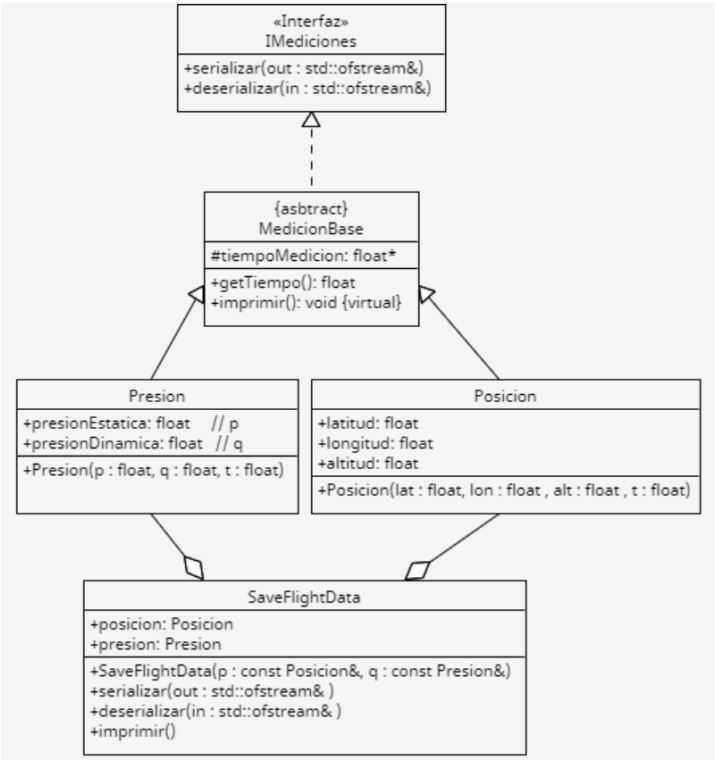

Figura 1: Diagrama de clase para el problema 1

Escriba el código que implementa este sistema teniendo en cuenta los siguientes puntos:

a. Dado que se desea que el tiempo en el que se tomó la medición pertenezca exclusivamente a la clase (no se quiere compartir el ownership), se debe utilizar un puntero std::unique_ptr para tiempoMedicion en la clase abstracta MediciónBase.

b. Escriba en el main una prueba simple para su código donde se empleen los siguientes valores:

Posicion posicion(-34.6f, -58.4f, 950.0f, 5.3);
Presion presion(101.3f, 5.8f, 6.1f)

E imprima el resultado utilizando los métodos imprimir de las clases. Puede sobreescribir los métodos que considere necesarios.

c. Al utilizar un puntero del tipo std::unique_ptr en la clase abstracta, este será parte de las clase Presion y Posicion. Debido a este tipo de puntero, el compilador evitará que estas clases puedan ser copiadas directamente, como es el caso de cuando se pasan a la clase SaveFlightData. Resuelva este problema sin utilizar std::move (utilice std::move para un puntaje parcial).

2. Escriba el código para las clases que representan:

a. Un punto, con su posición (x, y) y los correspondientes setters y getters.

b. Un círculo, con su posición, radio (r) y los correspondientes setters y getters.

c. Una elipse, con la posición de su centro, el semieje mayor (a), el semieje menor (b) y los correspondientes setters y getters.

d. Un rectángulo, con la posición de su vértice izquierdo inferior, su ancho, su largo y los correspondientes métodos setters y getters.

Cree una clase adicional denominada ProcesadorFigura que utilice especialización de plantilla para calcular el área de las figuras.
Escriba el código en el main que permita ejemplificar como calcular el área de cada una de las figuras mencionadas.

3. Escriba un código que permita crear el siguiente JSON:

```
{  "vec_doubles" : [1.3, 2.1, 3.2],    // Aquí sólo se utiliza el tipo de dato double
   "palabras" : ["Hola", "Mundo"], // Aquí sólo se utiliza el tipo de dato std::string
   "listas" : [   // Las listas sólo deben ser creadas con tipos de datos enteros
              [1, 2],
              [3, 4]
             ]
}
```

Para lograr esto, cree dos clases que colaboren para construir el JSON en partes separadas:

1. **Clase 1:** esta clase se encargará de generar los vectores de *double* y *std::string*, así como la matriz de enteros. Para ello, se debe utilizar un método para agregar los valores y las palabras a la instancia de la clase para luego procesarlas según el tipo de dato que se pasó.
   Deberá utilizar un **template** para poder trabajar con los distintos tipos de datos. En función del tipo, y **mediante el uso obligatorio de "**if constexpr", se deberán aplicar distintos métodos para procesar los datos adecuadamente.
2. **Clase 2:** Esta clase se encargará de asociar etiquetas (por ejemplo, "palabras") con su vector/matriz correspondiente (por ejemplo, ["Hola", "Mundo"]).
   Además, contendrá el método que finalmente construya el JSON completo y lo imprima por pantalla.

Cree un programa que utilice estas clases para crear y visualizar el JSON resultante. El formato de salida debe coincidir con el mostrado al inicio del enunciado.