PRÁCTICA 6

1. RESOLVER Y RESPONDER

1) Modelizar utilizando diagramas de clases (UML) los siguientes objetos. Defina atributos y métodos (para una funcionalidad mínima).
   a. Fecha.
   b. Automóvil.
   c. Ventana del sistema operativo Windows.
   d. Sala de cine.
2) Definir los tipos de atributos que puede tener una clase y su significado.
3) Definir los tipos de métodos que puede tener una clase y su significado.
4) Definir qué es un constructor y qué es un destructor en un lenguaje de programación orientado a objetos.

2. DISEÑAR E IMPLEMENTAR

5) Diseñe en UML e implemente en C++. En cada caso incluya prototipos para un constructor (o varios) y funciones miembro que permitan manipular cada uno de sus atributos. Por ejemplo, si la clase tiene el atributo `hora`, implementar las funciones `setearHora(unaHora)` y `mostrarHora()`. Esto hace que se puedan manipular de forma correcta los miembros de datos privados de la clase. Implemente las funciones pedidas e incluya otras funciones si es necesario. Pruebe el correcto funcionamiento de cada Clase, en un programa de C++.
   a. Una clase llamada `Tiempo` con atributos de tipo entero llamados `horas`, `minutos` y `segundos`.
   b. Una clase llamada `Complejo` con atributos en número de precisión doble llamados `real` e `imaginario`.
   c. Una clase llamada `Circulo` con atributos en número entero llamados `centro_x` y `centro_y` y otro atributo en número de precisión doble llamado `radio`.

6) Diseñe e implemente una clase llamada `Rectangulo` que tenga atributos en número de precisión doble llamados `largo` y `ancho`. La clase deberá tener una función miembro llamada `perímetro()` y otra función miembro llamada `area()` para calcular el perímetro y el área de un rectángulo. Implementar las funciones para manipular los atributos (asignarles valor y mostrar). Además, implementar una función miembro llamada `mostrarDatos()` que muestre el largo, ancho, perímetro y área de un rectángulo (utilizando los propios métodos de la clase).

7) Modifique la clase `Fecha` para que contenga un método que compare dos objetos de tipo fecha `Fecha`. El método lo recibe un objeto de clase `Fecha` y tiene como parámetro otro objeto de clase `Fecha`. Retorna verdadero si `fecha1` es menor a `fecha2` argumento:

```
fecha1.Menor(fecha2)
```

Una posible solución puede describirse como:

Algorithm 1: Función de comparación entre 2 fechas.

```
{
   Determina si una fecha es menor a otra según el siguiente procedimiento:
     Convierte cada fecha en un valor entero de la forma aaaammdd
     (esto puede lograrse usando la fórmula año*10000 + mes * 100 + día)
     Compara los enteros correspondientes para cada fecha
     El número entero más pequeño corresponde a la fecha menor.
   Devuelve verdadero o falso según corresponda
}
```

8) Diseñar e implementar en C++ la clase Pila de enteros.
9) Diseñar e implementar en C++ la clase Cola de enteros.
10) Diseñar e implementar en C++ una plantilla (template) para la Clase pila.
11) Diseñar e implementar en C++ una plantilla (template) para la Clase cola.

Nota: para compilar de forma correcta incluya el .h y el .cpp de la plantilla en el .cpp donde esté la función main(). Por ejemplo, en el .cpp donde se encuentra la función main() y se usan las funciones del template cola, se incluyen:

```
#include ”Cola.h”
#include ”Cola.cpp”
```