Práctica IV

Todos debéis entregar un único fichero [practica4.zip] antes de la finalización de la práctica. Al comienzo del mismo debéis incluir un comentario con vuestro nombre y el de vuestro compañero en caso de realizar la práctica en pareja.

Recordad que las prácticas se realizan en vuestra sala de grupo/individual (según la modalidad escogida) siguiendo el procedimiento habitual (compartir audio, webcam, escritorio y grabando la sesión). Y que podéis realizar consultas en internet, apuntes, campus, etc. No estando permitido herramientas de comunicación fuera del grupo de trabajo.

Cualquier incumplimiento supondrá la descalificación directa en la práctica.

*Apartado 1 [**4 puntos**]*: templatizar una clase de nombre ***Contenedor*** que encapsule un tipo de datos std::vector<T>. Dicha clase debe incorporar los constructores y métodos getter/setter que consideréis oportunos, así como un método ordenar que devuelva un std::vector<T> con los elementos del vector templatizado encapsulado en la clase ordenados mediante el método de la burbuja. Esta clase templatizada debe permitir al menos ordenar números enteros, números decimales, strings y personas por edad.

Debéis utilizar la siguiente declaración del tipo de datos Persona, completando la definición de sus métodos.

```
struct persona{
        int edad=0;
        string nombre;

        persona(int e,string n);
        bool operator >(const persona &per);
};
```

*Apartado 2 **[6 puntos]***: templatizar una clase de nombre ***Pila*** de manera que podamos utilizarla con distintos tipos de datos. Para ello, en lugar de utilizar punteros, vamos a encapsular en dicha plantilla un tipo de datos vector<T> y vamos a implementar los métodos push, pop e imprimir.

- Método push: debe añadir un elemento en nuestra pila.
- Método pop: debe eliminar un elemento en nuestra pila
- Método imprimir: debe mostrar por terminal el contenido completo de la pila.

A continuación debéis probar vuestra pila en la función principal del programa, para ello vais a declarar:

- Una estructura templatizada de nombre ***Nodo*** que encapsule:
    - std::string nombre;
    - int IDnodo;
    - T infoNodo;
- Una estructura de nombre ***Color*** que encapsule:
    - std::string nombre;
    - int R;
    - int G;
    - int B;

Además, debéis sobrecargar el operador << para poder mostrar por terminal objetos de tipo Nodo. Recordad que Nodo es un template, por lo que deberéis templatizar también la sobrecarga de << para indicar en el parámetro de entrada donde pasáis el nodo a mostrar por terminal el tipo templatizado que corresponda.

Una vez que tengáis todo esto implementado podréis, en vuestro programa principal, crear un objeto pilaColores y apilar, desapilar e imprimir el contenido de la pila por terminal de la manera:

```
Pila<Nodo<Color>> pilaColores;
pilaColores.push({"Nodo 1",101,{"Rojo",255,0,0}});
```

Probad también a usar una pila de enteros, de strings, de vectores que contengan doubles, y de vectores que contengan personas.

Como recordatorio: una pila es un TAD lineal LIFO (Last Input First Output) cuyo funcionamiento se asemeja, por ejemplo, a una pila de platos. Vamos colocando información (nodos o platos :) uno encima de otro y cuando queremos recuperar información de nuestro stack siempre lo hacemos por la parte alta de la pila (TOS, Top Of Stack)