Programación Orientada a Objetos - Parcial 1 - 6/10/2020 - Tema 1

Ejercicio 1

Realice una función llamada ...intercala(.....) que recibe como parámetros 2 punteros a vectores de enteros La función debe devolver un nuevo vector con los elementos intercalados de los vectores pasados como parámetro (por ejemplo si a=[2,4,5,1,6,3] y b=[10,20] se deberá obtener [2,10,4,20,5,1,6,3]. Utilice la función en un programa cliente.
NOTAS: No usar la clase vector: Usar solo notación de punteros (no usar operador [ ]).

Ejercicio 2

En una aplicación que gestiona los datos de un conjunto de estudiantes, hay una función para obtener los *n* mejores promedios para otorgarles una cierta beca:

```
void obtenerBeneficiados(int n, LogSystem log) {

    log.registrarMensaje("Cargando archivo de datos");
    vector<Alumnos> v = cargarDatos("alumnos.dat");

    log.registrarMensaje("Filtrando Insuficientes");
    for(Alumno &a : v)
        filtrarInsuficientes(a.notas);

    log.registrarMensaje("Recalculando promedios");
    for(Alumno &a : v)
        a.prom = calcularPromedio(a.notas);

    log.registrarMensaje("Ordenando por promedio");
    ordenarVector(v,comparaPorPromedio);

    log.registrarMensaje("Guardando los "+to_string( n )+" mejores");
    v.resize( n );
    guardarDatos("becarios.dat");

    log.finalizar();

}
```

La función recibe la cantidad de becarios que debe obtener, y un objeto para "loggear" sus acciones (registrar qué acciones realiza en cada paso mediante mensajes de texto).

Utilizando polimorfismo, diseñe dos clases: ***CoutLogger*** y ***VectorLogger***, para usar como 2do argumento de la función (sin modificarla):

- La primera debe escribir en pantalla los mensajes que recibe inmediatamente cuando los recibe (método *registrarMensaje*(...)), y no hacer nada más al finalizar.
- La segunda debe guardar los mensajes en un vector de strings a medida que los recibe, y mostrarlos todos juntos al finalizar (método *finalizar()*).

Escriba un programa cliente que permita al usuario elegir el mecanismo de logging e invoque a la función *obtenerBeneficiados* para encontrar los 10 mejores promedios. Para

que todo esto funcione correctamente, deberá hacer un pequeño cambio en el prototipo de la función *obtenerBeneficiados*, ¿cuál? y ¿por qué?

Ejercicio 3

a) En la clase Alumno: ya están completos los datos o atributos; proponga un constructor para iniciar sus datos: nombre, DNI y nota_final; un método para determinar la condición: 'R' o 'L' ('R' es Regular si nota_final>=4, 'L' es Libre si nota_final<4) y otros métodos que considere necesarios o adecuados para consultar los datos de la clase.
b) Proponga una clase Materia, que reutilice la clase Alumno mediante la relación que considere más adecuada. La clase Materia debe permitir definir el nombre de la materia, el nombre del profesor, y registrar varios alumnos. Proponga un constructor, y métodos para obtener la cantidad de regulares y de libres; y otros métodos que crea necesarios.
Solo proponga las clases y codifique sus métodos (no hacer programa cliente).

```
class Alumno {
    string nombre;
    int DNI, nota_final;
    . . .
  public:
    . . .
};
```

Programación Orientada a Objetos - Parcial 1 - 6/10/2020 - Tema 2

Ejercicio 1

a) Diseñe una función C++ **superaprom(..)** que tenga como parámetros la dirección de un arreglo de flotantes, y su longitud. La función debe devolver la cantidad de elementos del arreglo que superan el promedio de sus elementos.
b) Luego escriba un programa C++ que ingrese la cantidad de datos flotantes a ingresar, defina un arreglo dinámico de flotantes e ingrese los datos del arreglo.
c) El programa debe invocar a la función **superaprom**(...) para obtener cuantos datos superan el promedio de los datos ingresados y después definir un nuevo arreglo dinámico que solo contenga los datos que superan al promedio. Mostrar el arreglo que tiene datos mayores al promedio.
NOTAS: No usar la clase vector: Usar solo notación de punteros (no usar operador [ ]).

Ejercicio 2

En una aplicación que gestiona los datos de un conjunto de estudiantes, hay una función para obtener los *n* mejores promedios para otorgarles un cierta beca:

```
void obtenerBeneficiados(int n, ProgressIndicator &p) {
        p.comenzar(5);
        vector<Alumnos> v = cargarDatos("alumnos.dat");
        p.paso();
        for(Alumno &a : v)
                filtrarInsuficientes(a.notas);
        p.paso();
        for(Alumno &a : v)
                a.prom = calcularPromedio(a.notas);
        p.paso();
        ordenarVector(v,comparaPorPromedio);
        p.paso();
        v.resize(n);
        guardarDatos("becarios.dat");
        p.finalizar();
}
```

La función recibe la cantidad de becarios que debe obtener, y un objeto p que se encargará de informarle al usuario el progreso de la aplicación. Antes de comenzar a procesar los datos, con el método comenzar(5) se le indica a p que comienza el procesamiento, y cuantos pasos van a ser (argumento del método, en este ej 5). Luego, cada vez que la aplicación completa un paso, se invoca al método "paso()". Cuando el proceso se completa y todos los pasos están listos, se invoca al método "finalizar()".

Diseñe dos clases: ***TextualProgress*** y ***PorcentualProgress***, para usar como 2do argumento de la función (sin modificarla):

- La primera, en los método comenzar y paso, deberá mostrar el mensaje "Ejecutando paso I de N" (reemplazando I por el número de paso en que va, y N por el total); y el método finalizar, el mensaje "Ejecución finalizada.".

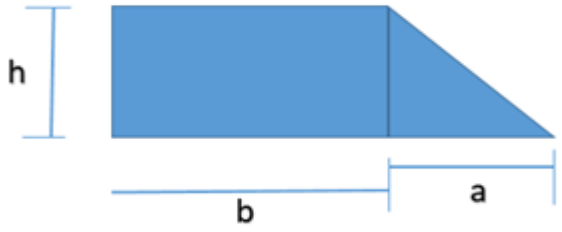
- La segunda, en los método comenzar y paso, deberá mostrar el mensaje "Progreso: X%" (reemplazando X por el porcentaje de avance actual según el paso en que está y la cantidad total); y el método finalizar, el mensaje "Proceso completado con éxito.".

Escriba un programa cliente que permita al usuario elegir entre ambas opciones e invoque a la función *obtenerBeneficiados* para encontrar los 10 mejores promedios. Responda, ¿exactamente qué problema/error tendría al compilar o ejecutar su programa si la función no recibiera el argumento log por referencia?

Ejercicio 3

Diseñe la clase Triangulo y la clase Rectangulo con los atributos base, altura. Proponga constructores para asignar datos y métodos para obtener el área en cada una de las clases.

Implemente la clase TrapecioRect que represente el trapecio rectángulo de la figura. Esta clase está compuesta por un triángulo y por un rectángulo, y debe plantear un constructor que asigne los datos a, b, h y un método que permita obtener el área del trapecio.



Escriba un programa que ingrese los datos a,b,h de una trapecio como el de la figura y muestre el área del trapecio a través de una instancia de la clase TrapecioRect.
Nota: Area de un rectangulo= base x altura; Area triangulo=base x altura /2