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):

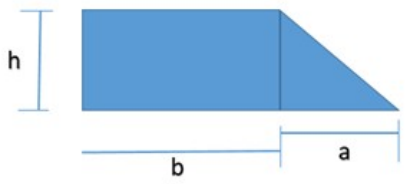
* La primera, en los método comenzar y paso, deberá mostrar el mensaje "Ejecutando paso I de N" (reemplazando I por el número de paso en que va, y N por el total); y el método finalizar, el mensaje "Ejecución finalizada.".

* 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 **p** 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

Programación Orientada a Objetos - RECUPERATORIO Parcial 1 - 29/10/2020

Ejercicio 1 (35pts)

Implemente la función: **tuple<int*,int> eliminaRepetidos(int* arreglo, int tam)**. La función recibe un puntero a un arreglo dinámico y la cantidad de datos (enteros) contenidos en el mismo. Debe eliminar los elementos repetidos, mediante el siguiente algoritmo:

1. crear un vector auxiliar con espacio suficiente para tantos elementos como el original
2. por cada elemento del vector original,
2.1 buscarlo en el auxiliar, si no está agregarlo
3. crear el vector definitivo, con espacio justo para la cant de elementos agregados al auxiliar
4. copiar los datos del auxiliar al definitivo
5. borrar los vectores original y auxiliar
6. retornar el definitivo y su nuevo tamaño en la tupla

Ej: si recibe [1,4,1,6,3,5,7,3,2,6,2,1,8,4] deberá retornar [1,4,6,3,5,7,8]). Use solo notación de punteros (no use el operador []) y no utilice la clase vector.

Ejercicio 2 (35pts)

a) Diseñe una clase para representar un viaje que guarde y permita consultar la siguiente información: ciudades de origen y destino, medio de transporte (un string, por ej "avión"), y distancia (en kms).

b) Implemente 3 clases que representen viajes en tres medios de transportes específicos: en avión, en auto y en colectivo. Cada una de estas clases debe permitir obtener el tiempo del viaje de la siguiente forma:

- en auto: t = distancia/110 + .5*(distancia%110) // se viaja a 110km/h, con un descanso de 30m cada 2h
- en avión: t = 2+distancia/900 // se viaja a 900km/h, pero hay que estar 2 horas antes en el aeropuerto
- en colectivo: t = distancia/90 // se viaja a 90km/h, sin esperas antes ni paradas

c) Implemente una función que reciba un vector de viajes y retorne el tiempo total (la suma).

d) Escriba un programa cliente que genere un vector con 3 viajes: de Rosario a Paraná en auto (200km), de Paraná a Buenos Aires en avión (400km), y de Buenos Aires a Rosario en colectivo (320km). El programa debe usar la función para obtener y luego mostrar el tiempo total.

Ejercicio 3 (30pts)

Diseñe una clase Alumno que tenga estos atributos: dni, nombre (apellido seguido de nombres), cantidad materias aprobadas y vector de notas (de materias aprobadas). Proponga además métodos para inicializar datos, calcular el promedio de las materias aprobadas y consultar todos sus datos. Proponga otros métodos y/o funciones que crea necesarios considerando que la porción de código de abajo debe funcionar.

```
void mostrar_alumno(Alumno a) {
  cout << "Nombre:" << a.ver_nombre() << endl;
  cout << "DNI:" << a.ver_dni() << endl;
  cout << "Notas:";
  for (int i=0; i<a.cantidad_de_notas(); ++i)
    cout << a[i] << " ";
}
int main() {
  Alumno a1(..), a2(..);
  ... // aca se cargarian las notas de cada uno
  cout << "El alumno con mejor promedio es:" << endl;
  if (a1>a2) mostrar_alumno(a1);
  else       mostrar_alumno(a2);
}
```

FICH-UNL. Programación Orientada a Objetos 2019. 1er Parcial. 24-09-2019. Tema B

Ejercicio 1 (30 pts) Implemente una clase Examen para modelar el enunciado completo de un examen, reutilizando la clase Ejercicio. La clase Examen debe tener: a) Un constructor que reciba el nombre de la materia y la fecha del examen, y métodos para consultar ambos datos. b) Un método para agregar un Ejercicio al Examen. c) Uno o más métodos que dado un número de Ejercicio, permitan consultar los datos del mismo. d) Un método CalcularCalificacion que reciba un vector<int> con las notas de un alumno en cada ejercicio y retorne su nota, calculada como porcentaje sobre la suma de los puntajes máximos de todos los ejercicios (esta suma no siempre es 100).

```
class Ejercicio {
   ...
public:
   Ejercicio(int puntaje,
             string enunciado,
             bool solo_libres);
   string VerEnunciado();
   int VerPuntajeMaximo();
};
```

Ejercicio 2. (25 pts) Escriba una clase llamada Hora con atributos para definir una hora (hs, min, seg). Proponga entre los métodos la sobrecarga del operador <= para comparar 2 horas y poder determinar la menor o igual (anterior o igual) y la mayor (hora posterior). Codifique solo la sobrecarga de <=.

Ejercicio 3 (30 pts) Un delivery de Pizzas las hace rectangulares y redondas. Cada pizza puede ser de un tipo diferente (muzzarella/palmitos/chocolate). Modele una clase base Pizza que permita guardar y consultar el tipo. Implemente las clases PizzaRectangular y PizzaRedonda, reutilizando la clase Pizza, que permitan definir en su constructor el tipo y sus dimensiones (ancho y alto para la rectangular, radio para la redonda). Implemente un método CalcularCosto(dólar) que reciba el precio del dólar y en base al mismo calcule el costo de una pizza. Para las cuadradas, el costo es: T x dólar x ancho x alto / 16. Para las redondas: como T x dólar x PI x radio^2 / 20. T vale 1.0 para la de muzzarella, 1.45 para la de palmitos, y 1.8 para la de chocolate. Implemente una función CostoPedido que reciba un arreglo/vector de Pizzas con todas las pizzas de un cierto pedido, y el valor del dólar al momento de cobrar el pedido; y sume y retorne el costo total del mismo.

Ejercicio 4 (15 pts) Explique) a) ¿Qué es una clase una clase abstracta? b) Antes de codificar, ¿cómo reconoce que dos clases pueden componerse en la relación de herencia? c) ¿Para qué se usan los operadores new y delete? Ejemplifique. d) Proponga un ejemplo para mostrar un arreglo de n elementos enteros a través de notación de punteros.

FICH-UNL. 1er Parcial POO - Tema B - 02/10/2018

Ej. 1) (30 Pts.) Diseñe una clase denominada ReciboSueldo, con los atributos dni, nombre, total, y un vector de items, donde se almacena la descripción y el monto (puede usar un struct), en caso que sea un descuento el valor será negativo. El atributo total, debe mantenerse actualizado; es decir, que cada vez que se cargue un nuevo item, debe recalcular y setear el total. a) Cree los métodos necesarios para cargar y consultar todos los atributos y haga uso del mismo desde el programa principal para cargar dos recibos de sueldo. b) Sobrecargue el operador +, de manera que aplicado a dos recibos, retorne la suma de ambos totales (notar que no retorna un recibo, sino solo el monto de la suma). c) Sobrecargue el operador de salida, para que muestre el dni, nombre, el subtotal de descuentos, el subtotal de haberes y el monto total a cobrar. Cree los métodos necesarios. d) Sobrecargue el operador de subscripción [ ] para que retorne un ítem. El operador [ ] debe además controlar que ese índice exista, y en caso contrario retornar un item cuya descripción sea la cadena vacía. Muestre todos los items y su correspondiente monto de una factura desde el programa principal.

Ej. 2 (20 pts) Escriba una función repite_vectord(...) que reciba un puntero a un arreglo de ints y su largo, y retorne un nuevo arreglo dinámico con el doble de elementos. La segunda parte debe contener los mismos elementos que la primera, pero en orden inverso. Ej. el arreglo [3, 2, 5] pasa a ser [3, 2, 5, 5, 3, 2].

Ej. 3 (30 pts). Programe una función llamada sumar_volumen que reciba un vector de diferentes tipos de cuerpos geométricos, todos clases hijas de una clase base Cuerpo, y retorna la suma de sus volumenes. Estas figuras cuentan con un método denominado calcular_volumen() que calcula y retorna el valor de volúmen del cuerpo correspondiente. Desde el programa principal cree un vector con una esfera(radio), un cilindro(radio, altura) y un cubo(lado), -implementando las tres clases hijas y usando los constructores tal como se indica- e invoque a la función previa para calcular el volumen total.

Nota: vol. de una esfera = 4/3 x pi x radio^3; vol. de un cubo = lado^3; vol. de un cilindro = pi x radio^2 x altura.

Ej. 4 (20 pts). a) ¿Qué diferencias hay entre una clase y un objeto? Ejemplifique. b) ¿En qué caso se debe sobrecargar el operador de asignación para que funcione correctamente? Ejemplifique. c) ¿Cuándo se invoca al constructor de un objeto? ¿y al destructor? d) Señale ventajas y desventajas de utilizar new y delete para gestionar la memoria.