Programación Orientada a Objetos - Parcial 1 - 27/09/22 - Tema A

Ej 1 (pts) a) Defina una clase "Estado" para guardar el estado de un personaje en un videojuego. El estado debe incluir el nivel actual (cuyo valor inicial será 1), una cantidad de vidas (cuyo valor inicial deberá ser recibido y asignado en el constructor de la clase), y una cantidad de puntos (que inicialmente debe estar en 0). La clase debe tener además un método para preguntar si el personaje está vivo (si la cantidad de vidas es mayor a 0), un método para incrementar en 1 el nivel, y otro para incrementar el puntaje en una cantidad que se reciba como argumento. Además, los operadores > y < para comparar dos estados y saber cuál está en un nivel más alto (y en caso de empate, cual tiene más puntos).

b) Defina una clase Personaje que tenga un nombre, una posición (coordenadas x e y) y un e

stado. Implemente los métodos que crea necesarios para cambiar la posición del personaje y para consultar y actualizar su estado. Utilice la clase en un programa cliente para crear dos personajes, incrementarles una cantidad aleatoria de veces el nivel a cada uno, sumarle una cantidad aleatoria de puntos a cada uno, y mostrar todos los datos del que tenga un mayor nivel (utilizando directa o indirectamente el operador sobrecargado).

Ej 2 (pts) a) Escriba una función que reciba tres punteros, uno apuntando al comienzo de un arreglo, otro apuntando a un elemento del arreglo (que puede ser cualquiera) y otro apuntando al final del arreglo (la dirección justo después del último elemento). La función debe generar un nuevo arreglo donde los elementos que estaban en el arreglo original después del señalado por el 2do puntero ahora estén antes; y los que estaban antes en el nuevo arreglo estén después. Por ejemplo, si el arreglo inicial es {1,2,3,4,5,6,7,8} y el 2do puntero apunta al 3, entonces el nuevo arreglo deberá ser {4,5,6,7,8,3,1,2}. Notar que la cantidad de elementos antes y después puede no ser igual.

b) Escriba un programa cliente que le permita al usuario ingresar un arreglo (el usuario indica el tamaño y los valores), y luego elegir un elemento para probar la función (un elemento o una posición para usar como 2do argumento). El programa debe mostrar luego de invocar a la función ambos arreglos.

Ej 3 (pts) Una empresa constructora tiene empleados para diferentes tipos de tareas: constructor, instalador y yesero. Los datos de cada empleado son el dni, nombre, apellido y sueldo básico. Declare las clases **Empleado**, **Constructor**, **Instalador** y **Yesero**. Las clases deben tener un método *MontoACobrar()* para calcular la remuneración mensual del trabajador. Puedes codificar todos los métodos adicionales que consideres necesarios. Los datos adicionales para el constructor son la cantidad de horas extras y los metros cubiertos, para el instalador son los días trabajados y para el yesero los metros cuadrados de revestimiento hechos. De acuerdo a la información, el cálculo los montos a cobrar correspondientes son:

Constructor = sueldo básico + 300 * horas extras + 100 * metros cubiertos

Instalador = sueldo básico + días trabajados * 3000

Yesero = sueldo básico + 200 * metros cuadrados

Codifique un programa cliente que tenga un único vector de 20 empleados (Constructor, Instalador y Yesero) y permita obtener el total de dinero que la administración deberá contar para pagarles el sueldo en un mes.

Ej 4 (pts) a) El encapsulamiento de miembros de una clase se hace con las clausulas private, protected o sin cláusula. Explique qué implicancias tiene el empleo de cada etiqueta. b) Si z y w son instancias de una clase en un programa, es posible ejecutar la siguiente porción de código: {.. if (z<w) cout<< z <<endl; else cout<< w <<endl; ….} ? Explique. c) Si se crea una variable dinámica través de un puntero p=new float; al ejecutar delete: explique qué se elimina del programa?

Programación Orientada a Objetos - Parcial 1 - 27/09/22 - Tema B

Ej 1 (30pts) a) Defina una clase *Vec2* para representar un punto o vector en 2 dimensiones (a partir de un par de coordenadas o componentes *x e y*). La clase debe tener una sobrecarga para la suma, otra para la resta, y un método para obtener el módulo. La suma y la resta se obtiene sumando/restando *x* con *x*, e *y* con *y*; el módulo como *raíz(x²+y²)*.
b) Implemente una clase *Proyectil* para representar un proyectil en un juego. El mismo debe tener como atributos una posición y un vector velocidad; y los valores iniciales para ambos deben ser recibidos en un constructor de la clase. La clase debe tener además dos métodos más: un método *mover* que modifique la posición sumandole la velocidad; y un método *colisiona* que reciba la posición de otro objeto y su radio (se supone que el otro objeto es redondo) y determine si la posición del proyectil colisiona con el otro objeto (si el módulo de la resta entre ambas posiciones es menor al radio).

Ej 2 (30pts) a) Escriba una función que reciba 4 punteros indicando: el comienzo de un arreglo, un elemento del arreglo (que puede ser cualquiera de ellos), otro elemento del arreglo (que debe ser posterior al del 2do argumento), y el final del arreglo (la dirección justo después del último elemento). La función debe generar un nuevo arreglo sin los elementos que estaban entre el 2do y el 3er puntero. Por ejemplo, si el arreglo inicial es *{11,22,33,44,55,66,77,88,99}*, y el 2do y 3er puntero apuntan al *33* y al *55* respectivamente, el nuevo arreglo debe ser *{11,22,66,77,88,99}.* b) Escriba un programa cliente para probar la función donde el usuario pueda cargar un arreglo indicando el tamaño y los elementos, y el programa pruebe la invocar a la función 10 veces, utilizando en cada prueba dos 2 posiciones del mismo generadas aleatoriamente para 2do y 3er argumento. El programa debe mostrar el nuevo arreglo en cada prueba.

Ej 3 (30pts) Una empresa tiene un conjunto de bienes que tienen un valor y deben ser amortizados (recuperar el costo a partir de los beneficios). Los bienes se clasifican en muebles, inmuebles y rodados. a) Defina una clase *Bien* que tenga como atributo el *valor* del mismo y un método *Amortizar* retorne el valor amortizado. b) Además codifique las clases *Mueble*, *Inmueble* y *Rodado* donde el cálculo para amortizar es el siguiente:
- En los bienes Mueble: *valor * (2022 - año de compra) * 0.05*
- En los bienes Inmueble: *0.0* // por ej un terreno, no se amortiza, por eso 0
- En los bienes Rodado: *valor * km / 10000*

De las fórmulas se desprende que además del valor, para los Muebles es necesario el año de compra, y para los Rodados los km. Puede codificar todos los métodos adicionales que consideres necesarios.
c) Cree un breve programa cliente donde declare un vector con 3 bienes, uno de cada tipo, y luego lo recorra mostrando la amortización de cada uno.

Ej 4 (10pts) Explique: a) Si *C1* y *C2* son candidatas a clases de un problema, como reconoce si hay relación de herencia entre *C1* y C2, y cuál hereda de cuál? ¿Cómo reconoce si hay composición, y cuál de las clases está contenida o cuál contiene a la otra? b) ¿Para qué sirven los constructores y destructores? ¿En qué casos es necesario implementarlos?. c) Considere un arreglo estático *int x[10]={100,102,104,106,108,110,112,114,116,118}* que comienza en la dirección de memoria *0x12a00*, si luego definimos *int *p=x*, ¿qué se obtendrá como salida de: *cout<<p<<" "<<p+2<<" "<<*(p+4)<<" "<<*(p+1)<<endl;*?