INSTRUCCIONES

- Es indispensable en la entrega de cada programa, que el código fuente sea completamente en idioma Español (a excepción de lo correspondiente a la API del lenguaje) [vigencia a partir del 6/Feb/2018].

TODOS LOS PROGRAMAS SOLICITEN AL USUARIO LOS DATOS DE ENTRADA NECESARIOS PARA LOS CÓMPUTOS.

Todos los ejercicios en adelante requieren de cumplir los Requerimientos de Valor Agregado en Código Fuente (hasta el requerimiento MM).

ES INDISPENSABLE para obtener calificación mayor a 5/100, el evitar solicitar los datos de entrada de usuario (el uso de "cin") dentro del ámbito de una clase.

IMPLEMENTAR LOS SIGUIENTES PROGRAMAS EN LENGUAJE C++ ANSI

1. Implementar y usar una clase llamada "Alumno" con al menos 3 atributos pasivos, uno de tipo cadena, otro de tipo entero y otro de tipo flotante. Obtener de entrada de consola los datos necesarios para dichos atributos. Implementar y usar los atributos activos (los métodos) fija() y dame(), correspondientes a cada atributo, es decir, los getters y setters. Al final del "main()" imprimir en consola los datos obtenibles de los atributos pasivos.

2. Implementar y usar en un mismo programa una clase llamada "Arbol" y una clase "Libro", ambas con al menos 3 atributos pasivos; utilizar en alguna cualquiera de las clases un atributo de tipo cadena, otro de tipo caracter, otro de tipo entero y otro de tipo flotante. Implementar y usar los atributos activos (los métodos) fija() y dame(), correspondientes a cada atributo, es decir, los getters y setters. Al final del "main()" imprimir en consola los datos obtenibles de los atributos pasivos.

3. Evolucionar el ejercicio 1, para implementar y usar en un mismo programa una clase llamada "Alumno" y una "Carrera"; colocar en la clase Carrera dos atributos, "idCarrera" y "nombre"; "idCarrera" sea de tipo entero; agregar a la clase "Alumno" el atributo "idCarrera". Implementar y usar los métodos fija() y dame(), correspondientes a cada atributo, es decir, los getters y setters. Al final del "main()" imprimir en consola los datos obtenibles de los atributos pasivos. Solicitar al usuario primero los datos del objeto "Carrera" y luego los datos del objeto "Alumno"; colocar en el "Alumno" el "idCarrera" mismo que haya sido capturado para el objeto "Carrera".

4. Evolucionar el ejercicio 2 para que en main() se solicite al usuario todos los datos necesarios para los dos objetos a utilizar; luego llamar a una subrutina pasándole como argumentos todos los datos necesarios para uno de los objetos; a continuación dicha subrutina, con los parámetros recibidos construya el objeto invocando un construtor sin parámetros que use datos por defecto; posteriormente, esta subrutina use los métodos "fija" para modificar los atributos del objeto; posterior a invocar a los métodos fija(), dentro de la citada subrutina que construyó al objeto, invocar a los métodos dame(), para finalmente mostrar la información del objeto.

5. En base a lo visto en carpeta "Ejemplos PRO\12 Programacion Orientada a Objetos\01 Arreglo de Objetos", implementar (o evolucionar lo implementado de) el ejercicio 51 de "Ejercicios PRO.pdf" a su forma orientado a objetos, donde la clase implementada cuente con atributos privados y sus correspondientes getters y setters.

6. En base a lo visto en la carpeta “Ejemplos PRO\12 Programacion Orientada a Objetos\02 Objetos compuestos\03 Atributos de tipo objeto”, reproducir por cuenta propia un programa que haga lo mismo que el ilustrado en el ejemplo, a decir, mostrar los datos por defecto del objeto, solicitar al usuario datos nuevos para todos los atributos e imprimir todos los datos del objeto, pero además, solicitar al usuario los datos para los objetos de tipo Fecha, que los atributos de la Fecha sean todos privados, e ingeniárselas para implementar y usar todos los métodos getters y setters propios de cada clase.

7. Evolucionar el ejercicio 3 para que se solicite a un usuario todos los datos necesarios para los dos objetos a utilizar; luego llamar a una subrutina pasándole como argumentos todos los datos previamente capturados de uno de los objetos; a continuación la subrutina con los parámetros recibidos construya el objeto solicitado mediante un constructor con parámetros, de modo que no se necesite invocar desde fuera del objeto a sus métodos “fija” para inicializar los atributos del objeto; invocar a los métodos “fija” al interior del constructor es lo adecuado; posterior a crear el objeto, invocar a los métodos dame() para mostrar la información del objeto. Elaborar un archivo de biblioteca (“.h”) que contenga la definición de la clase e incluir en el archivo “main.cpp” a dicho archivo.

8. Evolucionar el ejercicio 7, para que el atributo “Alumno.idCarrera” sea de tipo “Carrera*” (apuntador/referencia a instancia de tipo Carrera). Instanciar los objetos “Carrera” y “Alumno” usando el operador “new”. Colocar en la instancia de “Alumno” el puntero que haya sido generado para el objeto “Carrera”. Implementar una subrutina que reciba solo un puntero a “Alumno”; llamar a dicha subrutina desde “main()”; tal subrutina imprima todos los atributos tanto del objeto “Alumno” como del objeto “Carrera”. El programa no utilice variables globales. Nótese que es deseable y permitido modificar el nombre del atributo Alumno.idCarrera por Alumno.carrera.

9. Evolucionar el ejercicio 2 para declarar dos variables de tipo apuntador a objeto, una para cada tipo de objeto; llamar desde main a una subrutina que reciba parámetros por referencia; dicha subrutina se encargue de solicitar al usuario todos los datos de uno de los objetos, donde los parámetros que pasen por referencia sean los que correspondan a los atributos de dicho objeto; análogamente a la subrutina anterior, hacer otra subrutina para capturar los datos del segundo objeto; posterior a capturar todos los datos de ambos objetos, instanciar en main ambos objetos usando el operador new y, usando constructores con parámetros; a continuación, desde main llamar a una subrutina pasándole la referencia a uno de los objetos con la finalidad de imprimir ese objeto en la consola invocando a sus métodos dame(); hacer otra subrutina que reciba apuntador al segundo objeto para también imprimir sus datos en consola.

10. Completar el programa en carpeta "Ejemplos EDA\04 Apuntadores\03 El ABC con arreglos de objetos" para que la class cuente con todos los atributos privados; que la operación “new” para crear un objeto se realice exclusivamente hasta que el usuario solicite el alta de un registro; eliminar el atributo “libre”, donde la operación para alta de un registro encuentre un espacio libre buscando una celda donde haya un “nullptr” asignado; finalmente la operación “baja” sea la encargada de liberar la memoria del objeto que corresponda y asigne “nullptr” a la celda cuyo objeto haya sido eliminado (es aceptable que en lugar de asignar “nullptr” en la celda del objeto eliminado, que aplique la técnica de recorido de registros (la cual recorrería los apuntadores a objeto), en cuyo caso necesitaría una variable global que represente la cuenta de objetos que hayan sido instanciados); al terminar el programa, toda la memoria dinámica haya sido liberada adecuadamente usando “delete”.

11. Evolucionar el programa implementado en el ejercicio anterior, cumpliendo con lo solicitado pero ahora, incluir la capacidad de arreglos de apuntadores dinámicos tal como se ilustra en carpeta “Ejemplos EDA\04 Apuntadores\04 Arreglos dinamicos\01 C++\02 Agrandando arreglo de apuntadores”.

12. Evolucionar el programa implementado en el ejercicio anterior, conteniendo la class un constructor con parámetros encargado de inicializar todos los atributos, y que tales parámetros se obtengan del usuario; para la opción de baja de profesor se aplique la técnica de recorrido de registros (la cual recorrería los apuntadores a objeto); al terminar el programa, toda la memoria dinámica haya sido liberada adecuadamente usando “delete” o “delete[]” según corresponda. Posteriormente a entregar este ejercicio, entrenarse para lograr programar desde cero un programa como este, en máximo 60 minutos.