Enunciados examen

UNIVERSIDAD NEBRIJA

ALUMNO:

EPS

Asignatura: G0460009 Programación II

Curso: 2021/2022 Examen: Final Fecha: 23-05-2022
Semestre: 2º Convocatoria: Ordinaria

Parte Práctica (10 puntos; 70% nota final)

Se debe entregar los ficheros de vuestros ejercicios mediante la actividad en el campus antes de la finalización del mismo.

Tiempo: Hasta las 14.00.

Ejercicio 1 [1 punto]

Se debe entregar un único archivo ejercicio1.cpp

Realizar una función que reciba como parámetro un número entero positivo y devuelva el mayor número múltiplo de 3 inferior a dicho número. Realizar un programa que pida al usuario un número y haciendo uso de la función muestre dicho funcionamiento.

Ejercicio 2 [1 punto]

Se debe entregar un único archivo ejercicio2.cpp

Realizar una función que reciba por parámetro una cadena de texto y un número entero positivo n y devuelva la cadena de texto repetida n veces, por ejemplo funcion("hola", 3) devuelve "holaholahola". Se pide realizar un programa que muestre dicho funcionamiento.

Ejercicio 3 [2 puntos]

Se debe entregar un único archivo ejercicio3.cpp

1. Implementar una clase que permita almacenar matrícula y marca de un coche.
   - Debe comprobar que la matrícula tiene 4 números y 3 letras, en caso contrario, debe lanzar una excepción.
   - La marca es un texto libre.
2. Sobrecargar los operadores ">" "<" "==" y "!="
   - El operador ">" indicará si el primer coche es más viejo que el segundo (usando la matrícula)
   - El operador "<" indicará si el primero coche es más nuevo que el segundo (usando la matrícula)
   - El operador "==" indicará si son el mismo coche.
   - El operador "!=" indicará si son coches distintos.
     (se presupone que todos los coches son españoles)
3. Sobrecargar el operador "<<" para que muestra por pantalla *marca - matricula*
4. Se pide realizar un programa que muestre dicho funcionamiento.

Ejercicio 4 [6 puntos]

Entregar todos los archivos .cpp y .h que se consideren oportunos.

Realizar una clase *Persona* que contenga los siguientes atributos (2 puntos):

- nombre
- año de nacimiento.

Implementar:

- Constructor con parámetros para inicializar los atributos (no puede haber constructor sin parámetros). Se debe verificar que la fecha de nacimiento es posterior a 1900, en caso contrario se lanzará una excepción.
- Constructor copia.
- Destructor, que mostrará por pantalla *nombre ha muerto* , siendo *nombre* el nombre de la persona.
- Funciones *getter* y *setter* (las funciones *setter* deben hacer las comprobaciones oportunas)

Realizar una clase *Bombero* que deriva públicamente de *Persona*. Debe contener los siguientes atributos (1 punto)

- Fuegos apagados.
- Longitud del bíceps.

Implementar:

- Constructor con parámetros (no debe contener constructor sin parámetros, y verificación de que los datos son mayores que 0).
- Constructor copia.
- Funciones getter y setter oportunas

Realizar una clase Futbolista que deriva públicamente de Persona. Debe contener los siguientes atributos (1 punto)

- Goles en su carrera deportiva.
- Millones en el banco.

Implementar:

- Constructor con parámetros (no debe contener constructor sin parámetros, y verificación de que los datos son mayores o iguales que 0).
- Constructor copia.
- Funciones getter y setter oportunas

Realizar un programa que contenga un vector de punteros a Futbolistas y Bomberos y mostrar por pantalla (2 puntos)

- Su nombre y año de nacimiento de todas las personas del vector.
- Si son bomberos: la longitud de su bíceps.
- Si son futbolistas: los millones que tienen en el banco.

UNIVERSIDAD NEBRIJA

ALUMNO:

EPS

Asignatura: G0460004 Programación II

Curso: 2021/2022 Examen: Final Fecha: 24-06-2022
Semestre: 2º Convocatoria: Extraordinaria

Parte Práctica (10 puntos; 70% nota final)

Se deben entregar los ficheros de vuestros ejercicios mediante la actividad en el campus antes de la finalización del mismo.

Tiempo: 2 horas 30 minutos.

Se desea realizar un programa para gestionar una biblioteca, para ello se necesita.

Tres clases para los siguientes tipos de elementos

- Editorial
  - Nombre
  - País
  - Página web.
- Autor
  - Nombre y apellidos
  - Lengua madre
- Libro
  - Título
  - Año de publicación
  - Autor (puntero a Autor)
  - Editorial (puntero a Editorial)

Se desea realizar un programa que utilizando estas clases implemente las siguientes funcionalidades.

1. **Crear la clase Autor y crear en el main 2 punteros a autores (1 punto)**
2. **Crear la clase Editorial y crear en el main 1 puntero a editorial (1 punto)**
3. **Cear la clase Libro y crear en el main 2 libros para cada autor, y añadirlo a un vector de punteros a libro (1 punto)**
4. **Mostrar por pantalla los datos de los cuatro libros**. Para eso se deberá sobrecargar el operador "<<" de modo que se pueda hacer *cout << unlibro << "\n"* **(3 puntos)**

   Mostrará la siguiente información:

   ```
   Título: El título del libro.
   Año: El año de publicación
   Author: El nombre del autor
   Idioma: La lengua madre del autor.
   Editorial: El nombre de la editorial.
   País: El país de la editorial.
   ```

5. **Hacer una funcion *printAuthorData* que reciba como parámetros un autor y el vector con todos los libros. Mostrar por pantalla todos los datos de los 2 autores y todos sus libros.** Para ello se deberá sobrecargar el operador "<<" de modo que se pueda hacer *printAuthorData(autor1, libros);* **(4 puntos)**

   Mostrará por pantalla

   ```
   Autor: El nombre del autor
   Idioma: La lengua madre del autor
   Libros:
   - Título libro 1
   - Título libro 2
   - Título libro 3
   - etc.
   ```

CONTINÚA EN LA PÁGINA SIGUIENTE

Un ejemplo de función main podría ser el siguiente

```
int main()
{
    vector<shared_ptr<Libro>> libros;

    auto autor1 = make_shared<Autor>(Autor{"Miguel de Cervantes","Español"});
    auto autor2 = make_shared<Autor>(Autor{"William Shakespeare","Ingles"});


    auto editorial1 = make_shared<Editorial>(Editorial{"Alfaguara","España","www.alfaguara.es"});

    auto libro1 = make_shared<Libro>(Libro{"El quijote", "1605", autor1, editorial1});
    auto libro2 = make_shared<Libro>(Libro{"Novelas Ejemplares", "1613", autor1, editorial1});
    auto libro3 = make_shared<Libro>(Libro{"Romeo y Julieta", "1597", autor2, editorial1});
    auto libro4 = make_shared<Libro>(Libro{"Hamlet", "1603", autor2, editorial1});

    libros.push_back(libro1);
    libros.push_back(libro2);
    libros.push_back(libro3);
    libros.push_back(libro4);

    printAuthorData(autor1, libros);
    printAuthorData(autor2, libros);

    cout << *libro1;
    cout << *libro2;
    cout << *libro3;
    cout << *libro4;

    return 0;
}
```