Laboratorio 07

Herencia

Nombre: Gabriel Fernando Rodriguez Cutimbo

CUI: 20212157

Grupo: B

Repositorio GitHub:

https://github.com/gaco123/EPCC_CCII.git

1. Ejercicios

Resolver los siguientes ejercicios planteados:

1. Crear una clase Persona del cual tendrá métodos asignar una edad y nombre. Una segunda clase, alumno, tendrá que heredar este contenido y a través de esta clase poder asignar los datos de edad y nombre de los estudiantes.

   Código "Persona.h":

```
#ifndef PERSONA_H
#define PERSONA_H
#include <string>
#include <iostream>
using namespace std;

class Persona{
public:
        Persona();
        ~Persona();
        void dar_edad(int);
        void dar_nombre(string);
        int obtener_edad();
        string obtener_nombre();
private:
        int edad;
        string nombre;
};

#endif
```

Código "Persona.cpp":

```
#include "Persona.h"

Persona::Persona(){
}
Persona::~Persona(){
}
void Persona::dar_edad(int _edad){
	edad=_edad;
}
void Persona::dar_nombre(string _nombre){
	nombre=_nombre;
}
int Persona::obtener_edad(){
	return edad;
}
string Persona::obtener_nombre(){
	return nombre;
}
```

Código "Alumno.h":

```
#ifndef ALUMNO_H
#define ALUMNO_H
#include "Persona.h"

class Alumno : public Persona{
public:
	Alumno();
	~Alumno();
	void mostrar_EdadNombre();
private:
};

#endif
```

Código "Alumno.cpp":

```
#include "Alumno.h"

Alumno::Alumno(){
}
Alumno::~Alumno(){
}
void Alumno::mostrar_EdadNombre(){
	int e = obtener_edad();
	string n = obtener_nombre();
	cout<<"La edad de la persona es: "<<e<<" y su nombre es: "<<n;
}
```

Código "Lab_07.cpp":

```
#include <iostream>
#include <string>
#include "Persona.h"
#include "Alumno.h"
using namespace std;

int main (){
	Alumno x;
	x.dar_edad(20);
	x.dar_nombre("Pepe");
	x.mostrar_EdadNombre();

	return 0;
}
```

Funcionamiento:

```
La edad de la persona es: 20 y su nombre es: Pepe
```

2. Crear una clase Color que mantenga 3 valores (RGB). Una segunda clase Material, tendrá como información una variable de texto que describa algún material (Ejemplo: madera, vidrio, platico, etc.) Una tercera clase, Objetos, deberá de heredar contenido de ambas clases con la finalidad de describir diferentes objetos en cuanto a color y el material. (Ejemplo: mesa de color café y material de plástico)

Código "Color.h":

```
#ifndef COLOR_H
#define COLOR_H
#include <string>
using namespace std;

class Color{
public:
	Color();
	~Color();
	void dar_RGB(int,int,int);
protected:
	int R;
	int G;
	int B;
};

#endif
```

Código "Color.cpp":

```
#include "Color.h"

Color::Color(){
}
Color::~Color(){
}
void Color::dar_RGB(int r, int g, int b){
	R=r;
	G=g;
	B=b;
}
```

Código "Material.h":

```
#ifndef MATERIAL_H
#define MATERIAL_H
#include <string>
using namespace std;

class Material{
public:
	Material();
	~Material();
	void dar_Material(string);
protected:
	string material;
};

#endif
```

Código "Material.cpp":

```
#include "Material.h"

Material::Material(){
}
Material::~Material(){
}
void Material::dar_Material(string _material){
	material=_material;
}
```

Código "Objeto.h":

```
#ifndef OBJETO_H
#define OBJETO_H
#include "Color.h"
#include "Material.h"
```

```
#include <iostream>

class Objeto : public Color , public Material{
public:
	Objeto();
	~Objeto();
	void darnombre_Objeto(string);
	void describir();
private:
	string objeto;
};

#endif
```

Código "Objeto.cpp":

```
#include "Objeto.h"

Objeto::Objeto(){
}
Objeto::~Objeto(){
}
void Objeto::darnombre_Objeto(string _objeto){
	objeto=_objeto;
}
void Objeto::describir(){
	string temp;
	if(R==0 && G==0 && B==0){
		temp="negro";
	}
	else if(R==255 && G==255 && B==255){
		temp="blanco";
	}
	else if(R==255 && G==0 && B==0){
		temp="rojo";
	}
	else if(R==0 && G==255 && B==0){
		temp="verde";
	}
	else if(R==0 && G==0 && B==255){
		temp="azul";
	}
	else{
		temp="error";
	}
	cout<<objeto<<" de color "<<temp<<" y material de "<<material;
}
```

Código "Lab_07.cpp":

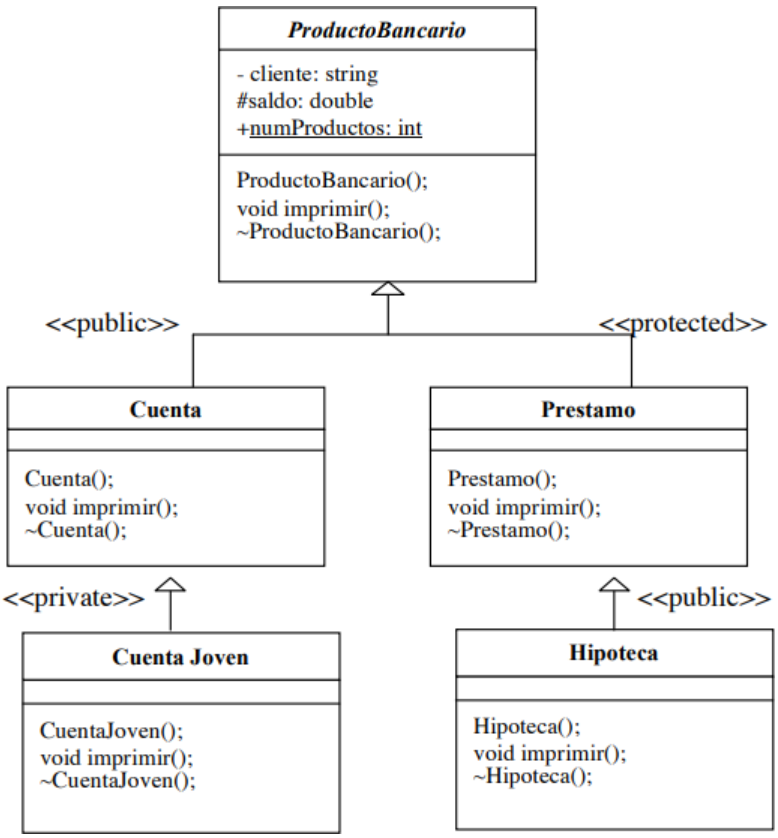
```
#include <iostream>
#include <string>
#include "Color.h"
```

```
#include "Material.h"
#include "Objeto.h"
using namespace std;

int main (){
	Objeto mesa;
	mesa.darnombre_Objeto("mesa");
	mesa.dar_Material("plastico");
	mesa.dar_RGB(255,0,0);
	mesa.describir();

	return 0;
}
```

Funcionamiento:

```
mesa de color rojo y material de plastico
```

3. Dada la siguiente jerarquía de herencia, indica la visibilidad de los atributos de la clase ProductoBancario en las clases CuentaJoven e Hipoteca.



Visibilidad de atributos para Cuenta Joven e Hipoteca:

***Cuenta Joven**

```
class Cuenta_Joven : private Cuenta{
public:
        CuentaJoven();
        void imprimir();
        ~CuentaJoven();
private:
        Cuenta(); //Heredado de Cuenta
        void imprimir(); //Heredado de Cuenta
        ~Cuenta(); //Heredado de Cuenta
        ProductoBancario(); //Heredado de Cuenta, pero antes, Cuenta heredo este atributo de ProductoBancario
        void imprimir(); //Heredado de Cuenta, pero antes, Cuenta heredo este atributo de ProductoBancario
        ~ ProductoBancario(); //Heredado de Cuenta, pero antes, Cuenta heredo este atributo de ProductoBancario
protected:
        //Para este ejercicio, va a depender si estos atributos están en private o protected en la clase ProductoBancario, ya que si están en private esto no se heredarían bajo ningún concepto mientras que si están en protected los siguientes atributos si se van a heredar.
        int cliente; //Heredado de Cuenta, pero antes, Cuenta heredo este atributo de ProductoBancario
        double saldo; //Heredado de Cuenta, pero antes, Cuenta heredo este atributo de ProductoBancario
        int numProductos; //Heredado de Cuenta, pero antes, Cuenta heredo este atributo de ProductoBancario
};
```

***Hipoteca**

```
class Hipoteca : public Prestamo{
public:
        Hipoteca();
        void imprimir();
        ~Hipoteca();
        Prestamo(); //Heredado de Prestamo
        void imprimir(); //Heredado de Prestamo
        ~Prestamo(); //Heredado de Prestamo
protected:
        ProductoBancario(); //Heredado de Prestamo, pero antes, Prestamo heredo este atributo de ProductoBancario
        void imprimir(); //Heredado de Prestamo, pero antes, Prestamo heredo este atributo de ProductoBancario
        ~ ProductoBancario(); //Heredado de Prestamo, pero antes, Prestamo heredo este atributo de
```

```
ProductoBancario

        //Para este ejercicio, va a depender si estos atributos están en private o protected en la clase ProductoBancario, ya que si están en private esto no se heredarían bajo ningún concepto mientras que si están en protected los siguientes atributos si se van a heredar.
        int cliente; //Heredado de Prestamo, pero antes, Prestamo heredo este atributo de ProductoBancario
        double saldo; //Heredado de Prestamo, pero antes, Prestamo heredo este atributo de ProductoBancario
        int numProductos; //Heredado de Prestamo, pero antes, Prestamo heredo este atributo de ProductoBancario
};
```

4. Escribe una clase de nombre ClaseDisco, que herede de la clase ClaseMultimedia los atributos y métodos definidos por usted. La clase ClaseDisco tiene, aparte de los elementos heredados, un atributo más también definido por usted. Al momento de imprimir la información debe mostrase por pantalla toda la información.

Código "ClaseMultimedia.h":

```
#ifndef CLASEMULTIMEDIA_H
#define CLASEMULTIMEDIA_H
#include <string>
#include <iostream>
using namespace std;

class ClaseMultimedia{
public:
        ClaseMultimedia(bool, string);
        ~ClaseMultimedia();
private:
        bool interactiva;
        string titulo;
};

#endif
```

Código "ClaseMultimedia.cpp":

```
#include "ClaseMultimedia.h"

ClaseMultimedia::ClaseMultimedia(bool _interactiva, string _titulo){
        interactiva=_interactiva;
        titulo=_titulo;
        cout<<"Se introdujeron los valores de ("<<interactiva<<", "<<titulo<<") en la ClaseMultimedia\n";
}
ClaseMultimedia::~ClaseMultimedia(){
}
```

Código "ClaseDisco.h":

```
#ifndef CLASEDISCO_H
#define CLASEDISCO_H
#include "ClaseMultimedia.h"

class ClaseDisco: public ClaseMultimedia{
public:
	ClaseDisco(bool,string,string);
	~ClaseDisco();
private:
	string calidad;
};

#endif
```

Código "ClaseDisco.cpp":

```
#include "ClaseDisco.h"

ClaseDisco::ClaseDisco(bool _interactiva, string _titulo, string _calidad):ClaseMultimedia(_interactiva,_titulo){
	calidad=_calidad;
	cout<<"Se introdujo el valor de ("<<calidad<<") en la ClaseDisco\n";
}
ClaseDisco::~ClaseDisco(){
}
```

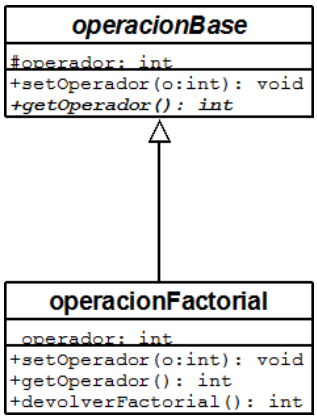
Código "Lab_07.cpp":

```
#include <iostream>
#include <string>
#include "ClaseMultimedia.h"
#include "ClaseDisco.h"
using namespace std;

int main (){
	//Permite ingresar 3 valores (bool,string,string) que tienen el nombre de (interactiva,titulo,calidad)
	//interactiva y titulo pertenecen a "ClaseMultimedia"
	 //calida pertenece a "ClaseDisco"
	 ClaseDisco L(false,"Parque Jurasico 3","Buena");

	 return 0;
}
```

Funcionamiento:

```
Se introdujeron los valores de (0, Parque Jurasico 3) en la ClaseMultimedia
Se introdujo el valor de (Buena) en la ClaseDisco
```

5. Escribe un programa que implemente la siguiente jerarquía de clases



Código "operacionBase.h":

```
#ifndef OPERACIONBASE_H
#define OPERACIONBASE_H

class operacionBase{
public:
	operacionBase();
	~operacionBase();
	void setOperador(int);
	int getOperador();
private:
	int operador;
};

#endif
```

Código "operacionBase.cpp":

```
#include "operacionBase.h"

operacionBase::operacionBase(){
}
operacionBase::~operacionBase(){
}
void operacionBase::setOperador(int _operador){
	operador=_operador;
}
int operacionBase::getOperador(){
	return operador;
}
```

Código "operacionFactorial.h":

```
#ifndef OPERACIONFACTORIAL_H
```

```
#define OPERACIONFACTORIAL_H
#include "operacionBase.h"

class operacionFactorial: public operacionBase{
public:
	operacionFactorial();
	~operacionFactorial();
	int devolverFactorial();
};

#endif
```

Código "operacionFactorial.cpp":

```
#include "operacionFactorial.h"

operacionFactorial::operacionFactorial(){
}
operacionFactorial::~operacionFactorial(){
}
int operacionFactorial::devolverFactorial(){
	int temp=1;
	for(int i=getOperador(); i>1; i--){
		temp=temp*i;
	}
	return temp;
}
```

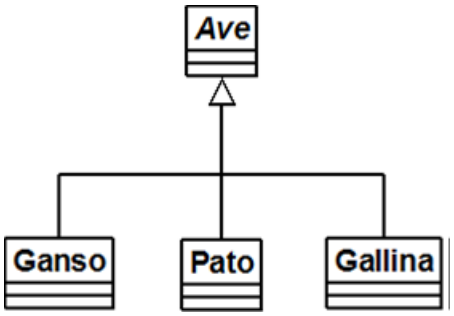
Código "Lab_07.cpp":

```
#include <iostream>
#include "operacionBase.h"
#include "operacionFactorial.h"
using namespace std;

int main (){
	int n = 10;
	operacionFactorial a;
	a.setOperador(n);
	cout<<"El factorial de "<<n<<" es: "<<a.devolverFactorial();

	return 0;
}
```

Funcionamiento:

```
El factorial de 10 es: 3628800
```

6. Escribe un programa que implemente la siguiente jerarquía de clases, Debe implementar aquellos atributos y métodos necesarios para que se pueda ejecutar el siguiente programa:



Código "Ave.h":

```
#ifndef AVE_H
#define AVE_H
#include <string>
#include <iostream>
using namespace std;

class Ave{
public:
	Ave();
	Ave(bool,string,string,string,string);
	~Ave();
	void mostrar_Vals();
protected:
	bool es_volador;
	string sonido;
	string tam_pico;
	string color_plumas;
	string tam_patas;
};

#endif
```

Código "Ave.cpp":

```
#include "Ave.h"

Ave::Ave(){
	sonido={};
	es_volador={};
	tam_pico={};
	color_plumas={};
	tam_patas={};
}
Ave::Ave(bool _es_volador, string _sonido, string _tam_pico, string _color_plumas, string _tam_patas){
```

```
	sonido=_sonido;
	es_volador=_es_volador;
	tam_pico=_tam_pico;
	color_plumas=_color_plumas;
	tam_patas=_tam_patas;
}
Ave::~Ave(){
}
void Ave::mostrar_Vals(){
	string temp{};
	if(es_volador==true){
		temp="Si";
	}
	else{
		temp="No";
	}
	cout<<"Sonido?: "<<sonido<<"\n";
	cout<<"Vuela?: "<<temp<<"\n";
	cout<<"Tamaño del pico?: "<<tam_pico<<"\n";
	cout<<"Color de las plumas?: "<<color_plumas<<"\n";
	cout<<"Tamaño de las patas?: "<<tam_patas<<"\n";
}
```

Código "Ganso.h":

```
#ifndef GANSO_H
#define GANSO_H
#include "Ave.h"

class Ganso: public Ave{
public:
	Ganso();
	Ganso(bool,string,string,string,string);
	~Ganso();
};

#endif
```

Código "Ganso.cpp":

```
#include "Ganso.h"

Ganso::Ganso(){
	sonido="On!, On!";
	es_volador=true;
	tam_pico="mediano";
	color_plumas="blancas";
	tam_patas="grandes";
}
Ganso::Ganso(bool _es_volador, string _sonido, string _tam_pico, string _color_plumas, string _tam_patas)
:Ave(_es_volador, _sonido, _tam_pico, _color_plumas, _tam_patas){
}
Ganso::~Ganso(){
}
```

Código "Pato.h":

```
#ifndef PATO_H
#define PATO_H
#include "Ave.h"

class Pato: public Ave{
public:
	Pato();
	Pato(bool,string,string,string,string);
	~Pato();
};

#endif
```

Código "Pato.cpp":

```
#include "Pato.h"

Pato::Pato(){
	sonido="Cuack!";
	es_volador=true;
	tam_pico="grande";
	color_plumas="verdes y marrones";
	tam_patas="medianas";
}
Pato::Pato(bool _es_volador, string _sonido, string _tam_pico, string _color_plumas, string _tam_patas)
:Ave(_es_volador, _sonido, _tam_pico, _color_plumas, _tam_patas){
}
Pato::~Pato(){
}
```

Código "Gallina.h":

```
#ifndef GALLINA_H
#define GALLINA_H
#include "Ave.h"

class Gallina: public Ave{
public:
	Gallina();
	Gallina(bool,string,string,string,string);
	~Gallina();
private:
};

#endif
```

Código "Gallina.cpp":

```
#include "Gallina.h"

Gallina::Gallina(){
```

```
        sonido="Kikiri, ki!";
        es_volador=false;
        tam_pico="pequeño";
        color_plumas="marrones, blancas o negras";
        tam_patas="cortas";
}
Gallina::Gallina(bool _es_volador, string _sonido, string _tam_pico, string _color_plumas, string _tam_patas)
:Ave(_es_volador, _sonido, _tam_pico, _color_plumas, _tam_patas){
}
Gallina::~Gallina(){
}
```

Código “Lab_07.cpp”:

```
#include <iostream>
#include <string>
#include <clocale>//Cáracteres Español
#include "Ave.h"
#include "Ganso.h"
#include "Pato.h"
#include "Gallina.h"
using namespace std;

int main (){
        setlocale(LC_CTYPE, "Spanish");//Cáracteres Español

        cout<<"Ave A\n";
        Ave a(true,"Onk!","grande","marrones","largas");
        a.mostrar_Vals();
        cout<<"\n";

        Ganso b;
        cout<<"Ganso B\n";
        b.mostrar_Vals();
        cout<<"\n";

        Pato c;
        cout<<"Pato C\n";
        c.mostrar_Vals();
        cout<<"\n";

        Gallina d;
        cout<<"Gallina D\n";
        d.mostrar_Vals();
        cout<<"\n";

        return 0;
}
```

Funcionamiento:

```
Ave A
Sonido?: Onk!
Vuela?: Si
Tamaño del pico?: grande
Color de las plumas?: marrones
Tamaño de las patas?: largas

Ganso B
Sonido?: On!, On!
Vuela?: Si
Tamaño del pico?: mediano
Color de las plumas?: blancas
Tamaño de las patas?: grandes

Pato C
Sonido?: Cuack!
Vuela?: Si
Tamaño del pico?: grande
Color de las plumas?: verdes y marrones
Tamaño de las patas?: medianas

Gallina D
Sonido?: Kikiri, ki!
Vuela?: No
Tamaño del pico?: pequeño
Color de las plumas?: marrones, blancas o negras
Tamaño de las patas?: cortas
```

2. Entregables

Al final estudiante deberá:

2.1. Compactar el código elaborado y subirlo al aula virtual de trabajo. Agregue sus datospersonales como comentario en cada archivo de código elaborado.

2.2. Elaborar un documento que incluya tanto el código como capturas de pantalla de laejecución del programa. Este documento debe de estar en formato PDF.

2.3. El nombre del archivo (comprimido como el documento PDF), será su LAB07_GRUPO_A/B/C_CUI_1erNOMBRE_1erAPELLIDO.

(Ejemplo: LAB07_GRUPO_A _2022123_PEDRO_VASQUEZ).

2.4. Debe remitir el documento ejecutable con el siguiente formato:

LAB07_GRUPO_A/B/C_CUI_ EJECUTABLE_1erNOMBRE_1erAPELLIDO

(Ejemplo: LAB07_GRUPO_A_EJECUTABLE_2022123_PEDRO_VASQUEZ).

En caso de encontrarse trabajos similares, los alumnos involucrados no tendrán evaluación y serán sujetos a sanción.