Reporte de Laboratorio 2
Herencia, polimorfísmo y sobrecargas en C++

Dunia Barahona - B40806

11 de septiembre de 2016

Índice

---

1. Código

1.1. Clase Base: Figura

Figura.h

```
#ifndef FIGURA_H
#define FIGURA_H

#include <cstdlib>
#include <iostream>
#include "math.h"
#include "string"

using namespace std;

class Figura{ //Clase base
public:
        string color; // atributos...
        string nombre;

        Figura();
```

```
        Figura(string nombre, string color);
        virtual ~Figura(); //destructor...

        virtual double area(); //virtuales...
        virtual double pmt();

        virtual void operator~(); //sobrecarga de operadores...
        virtual void operator!();
};
#endif /* FIGURA_H */
```

Figura.cpp

```
#include "Figura.h"

Figura::Figura() {  //constructores
}
Figura::Figura(string nombre, string color) {
        this->nombre = nombre;
        this->color = color;
}
Figura::~Figura() {  //destructor
}

//virtuales: que se reimplementaran en las clases derivadas
double Figura::area() {
        cout << "Metodo para calcular el area de la figura"<<
        this->nombre<< "debe implementarse en clases derivadas." << endl;
        return 0.0;
}
double Figura::pmt() {
        cout << "Metodo para calcular el perimetro de la figura"<<
        this->nombre<< "debe implementarse en clases derivadas." << endl;
        return 0.0;
}
void Figura::operator ~() {  //imprime los atributos del objeto.
        cout << "Nombre: "<< this->nombre << endl;
        cout << "Color:  " <<this->color << endl;
}
void Figura::operator !() {  //imprime el area y perimetro de la figura
        cout << "Area:   "<< this->area() << endl;
        cout << "PerÃmetro: "<< this->pmt() << endl;
        cout << endl;
}
```

1.2. Clase derivada: Círculo

Circulo.h

```
#ifndef CIRCULO_H
#define CIRCULO_H

#include "Figura.h"

//Clase 'Circulo' que hereda publicamente de la clase 'Figura'
class Circulo : public Figura{
public:
        double radio; //atributo propio...
        Circulo();
        Circulo(string nombre, string color, double radio);
        virtual ~Circulo(); //destructor...

        virtual double area();  //reimplementados...
        virtual double pmt();

        virtual void operator~(); //sobrecarga de operadores...
        virtual void operator!();
};
#endif /* CIRCULO_H */
```

Circulo.cpp

```
#include "Circulo.h"

Circulo::Circulo() {
}
Circulo::Circulo(string nombre, string color, double radio) {
        this->nombre =nombre;
        this->color =color;
        this->radio =radio;
}
Circulo::~Circulo() {
}
double Circulo::area() {
        double r= this->radio;
        double a= 3.1415*(r*r);
        return a;
}
double Circulo::pmt() {
        double r= this->radio;
        double p= 2*3.1415*r;
        return p;
}
```

```
void Circulo::operator ~() {
        cout << "Nombre: "<<this->nombre << endl;
        cout << "Color:  " <<this->color << endl;
        cout << "Radio:  "<<this->radio << " cm"<<endl;
}
void Circulo::operator !() {
        cout << "Area:   "<<this->area()<<endl;
        cout << "Perimetro: "<<this->pmt() <<endl;
        cout << endl;
}
```

1.3. Clase derivada: Cuadrado

Cuadrado.h

```
#ifndef CUADRADO_H
#define CUADRADO_H

#include "Figura.h"

//Clase 'Cuadrado' que hereda publicamente de la clase 'Figura'
class Cuadrado : public Figura {
public:
        double lado; //atributo propio...
        Cuadrado();
        Cuadrado(string nombre, string color, double lado);
        virtual ~Cuadrado(); //destructor...

        virtual double area(); //reimplementados...
        virtual double pmt();

        virtual void operator~(); //sobrecarga de operadores...
        virtual void operator!();
};
#endif /* CUADRADO_H */
```

Cuadrado.cpp

```
#include "Cuadrado.h"

Cuadrado::Cuadrado() {
}
Cuadrado::Cuadrado(string nombre, string color, double lado) {
        this->nombre =nombre;
        this->color =color;
        this->lado =lado;
}
```

```
Cuadrado::~Cuadrado() { //destructor
}
double Cuadrado::area() {
        double l= this->lado;
        double a= l*l;
        return a;
}
double Cuadrado::pmt() {
        double l= this->lado;
        double p= 4*l;
        return p;
}
void Cuadrado::operator ~() {
        cout << "Nombre: "<<this->nombre << endl;
        cout << "Color:  "<<this->color << endl;
        cout << "Longitud de lado: "<<this->lado << " cm"<<endl;
}
void Cuadrado::operator !() {
        cout << "Area:   "<<this->area() << endl;
        cout << "Perimetro: "<<this->pmt() << endl;
        cout << endl;
}
```

1.4. Clase derivada: Triángulo

Triangulo.h

```
#ifndef TRIANGULO_H
#define TRIANGULO_H

#include "Figura.h"

// Clase 'Triangulo' que hereda publicamente de la clase 'Figura'
class Triangulo : public Figura{
public:
        double lado_1; //atributos propios: lados del triangulo...
        double lado_2;
        double lado_3;
        Triangulo();
        Triangulo(string nombre, string color, double lado_1,
        double lado_2, double lado_3);
        virtual ~Triangulo(); //destructor...

        double semip(); //propio...
        virtual double area(); //reimplementados...
        virtual double pmt();
```

```
        virtual void operator~(); //sobrecarga de operadores...
        virtual void operator!();
};
#endif /* TRIANGULO_H */
```

Triangulo.cpp

```
#include "Triangulo.h"

Triangulo::Triangulo() {
}
Triangulo::Triangulo(string nombre, string color, double lado_1,
double lado_2, double lado_3) {
        this->nombre =nombre;
        this->color =color;
        this->lado_1 =lado_1;
        this->lado_2 =lado_2;
        this->lado_3 =lado_3;
}
Triangulo::~Triangulo() { //destructor
}
double Triangulo::area() {
        double s= this->semip();
        double a= this->lado_1;
        double b= this->lado_2;
        double c= this->lado_3;
        double ar= sqrt(s*(s-a)*(s-b)*(s-c));
        return ar;
}
double Triangulo::semip() { //calcula el semiperimetro del triangulo.
        double a= this->lado_1;
        double b= this->lado_2;
        double c= this->lado_3;
        double sp= (a+b+c)/2;
        return sp;
}
double Triangulo::pmt() {
        double p= this->lado_1 + this->lado_2 + this->lado_3;
        return p;
}
void Triangulo::operator ~() {
        cout << "Nombre: "<<this->nombre << endl;
        cout << "Color:  " <<this->color << endl;
        cout << "Longitud del lado 1: "<<this->lado_1 <<" cm"<< endl;
        cout << "Longitud del lado 2: "<<this->lado_2 <<" cm"<<endl;
        cout << "Longitud del lado 3: "<<this->lado_3 <<" cm"<<endl;
}
void Triangulo::operator !() {
```

```
        cout << "Area:    "<<this->area() <<endl;
        cout << "Perimetro: "<<this->pmt() <<endl;
        cout << endl;
}
```

1.5. main

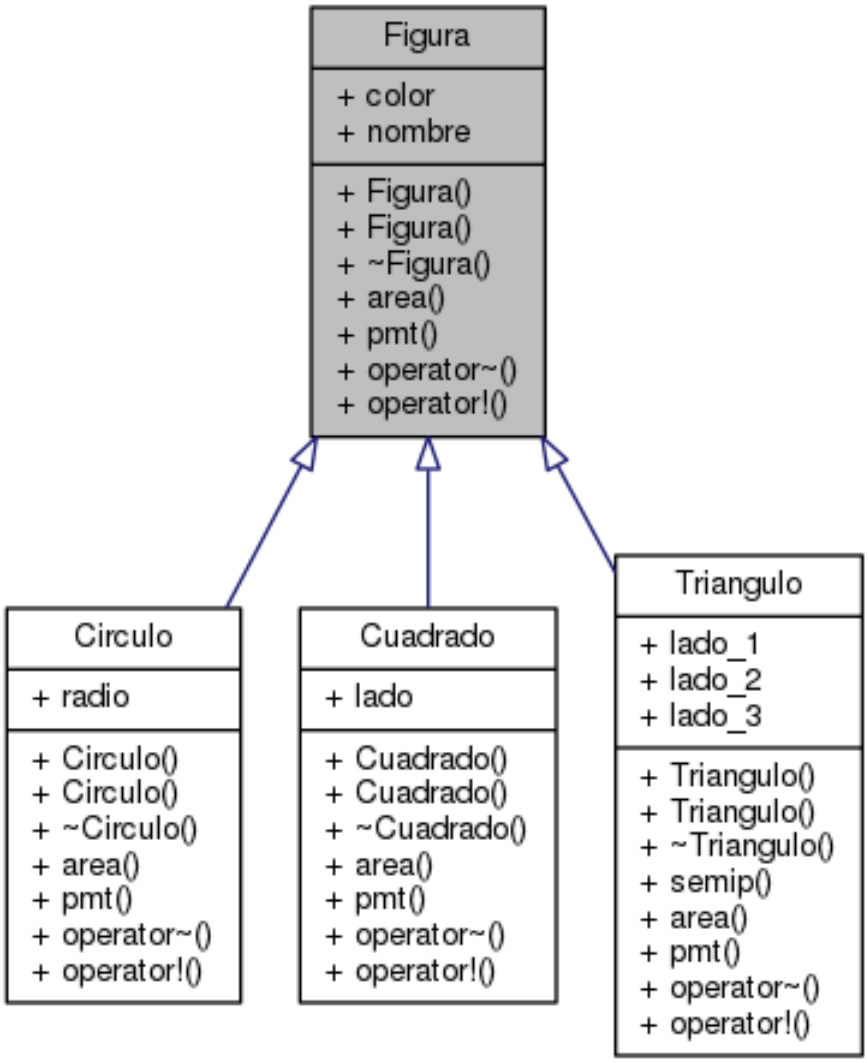
```
#include "Figura.h"
#include "Circulo.h"
#include "Cuadrado.h"
#include "Triangulo.h"

int main(int argc, char** argv) {
        // crea objeto de tipo Figura llamado 'F' y lo inicializa
        Figura F= Figura("cualquiera", "rojo");
        ~F;
        !F;
        Circulo cc= Circulo("circulo", "azul", 4.2);
        ~cc;
        !cc;
        Cuadrado cd= Cuadrado("cuadrado", "verde", 13.5);
        ~cd;
        !cd;
        Triangulo t= Triangulo("triangulo", "amarillo", 11, 7.5, 11);
        ~t;
        !t;

        return 0;
}
```

2. Diagrama de clases

Figura 1: Diagrama UML de la clase `Figura`



3. Conclusiones

1. Una clase derivada hereda los atributos y métodos públicos de la clase base.
2. Los métodos virtuales de la clase base son los que se pueden reimplementar en las clases derivadas.
3. La sobrecarga de operadores facilita la implementación de métodos.