Reporte de Laboratorio 2

Isacc Gómez Sánchez- B32919
José Fernando González Salas - B43023

11 de septiembre de 2016

Índice

---

1. Introducción

En el siguiente reporte, se presenta un programa en el cual se utilizan clases base y clases derivadas en orden con la programación orientada a objetos del lenguaje de C++. Además se realizan cálculos por medio de métodos virtuales y se le asignan valores a atributos propios de cada clase.

1.1. Objetivos

- Escribir un programa utilice clases bases y clases derivadas en C++
- Sobrecargar los operadores «˜» y «!» para realizar funciones específicas de impresión.
- Familiarizarse con los métodos virtuales de las clases.

2. Código

2.1. Archivo de encabezados

En este archivo encontramos los paquetes necesarios incluídos en el código para su correcto funcionamiento, además de la definición de los métodos y clases del programa.

```
// Figuras.h
#ifndef FIGURAS_H
#define FIGURAS_H

#include <cstdlib>
#include <string>
#include <ctime>
#include <iostream>

using namespace std;

// Clase base
class Figuras
{
    public:
        Figuras();
        Figuras(string nombre, string color);
        Figuras(const Figuras& orig);
        virtual ~Figuras();

        string nombre;
        string color;


   virtual void calcularArea();
   virtual void calcularPerimetro();

   virtual void operator~();
   virtual void operator!();
};

// Clases derivadas de Figuras
class Cuadrado: public Figuras
{
   public:
   Cuadrado();
        Cuadrado(string nombre, string color, double lado);
        Cuadrado(const Cuadrado& orig);
        virtual ~Cuadrado();

        string nombre;
        string color;
        double lado;
        double area;
        double perimetro;

        virtual void calcularArea(double lado);
        virtual void calcularPerimetro(double lado);
        virtual void operator~();
   virtual void operator!();
};

class Triangulo: public Figuras
```

```
{
   public:
        Triangulo();
        Triangulo(string nombre, string color, double base, double altura, double
            hipotenusa);
        Triangulo(const Triangulo& orig);
        virtual ~Triangulo();

        string nombre;
        string color;
        double base;
        double altura;
        double hipotenusa;
        double area;
        double perimetro;

        virtual void calcularArea(double base, double altura);
        void calcularPerimetro(double base, double altura, double hipotenusa);
        virtual void operator~();
   virtual void operator!();
};

class Circulo: public Figuras
{
   public:
      Circulo();
        Circulo(string nombre, string color, double radio);
        Circulo(const Circulo& orig);
        virtual ~Circulo();

        string nombre;
        string color;
        double radio;
        double area;
        double perimetro;

        const double PI = 3.141592653589793238463;
        virtual void calcularArea(double radio);
        virtual void calcularPerimetro(double radio);
        virtual void operator~();
   virtual void operator!();
};

#endif // FIGURAS_H
```

2.2. Archivo Figuras.cpp

Se les da definición y cuerpo a los constructores de las clases, tanto vacíos como sobrecargados. Además de indicar el cuerpo de las funciones de cálculo de área y perímetro de las figuras.

```
// Figuras.cpp
#include "Figuras.h"
```

```
Figuras::Figuras() {};

Figuras::Figuras(string nombre, string color) {
    this->nombre = nombre;
    this->color = color;
};

Figuras::Figuras(const Figuras& orig){};

Figuras::~Figuras(){};

void Figuras::calcularArea(){};

void Figuras::calcularPerimetro() {};

void Figuras::operator~(){
   cout << this->nombre << this->color << endl;
};

void Figuras::operator!(){
    cout << "0" << endl;
    cout << "0" << endl;
};

           /*  Clases derivadas */


//Cuadrado

Cuadrado::Cuadrado() {}

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

Cuadrado::Cuadrado(const Cuadrado& orig) {};

Cuadrado::~Cuadrado() {};

void Cuadrado::calcularArea(double lado) {
        this->area=(lado * lado);
   return;
};

void Cuadrado::calcularPerimetro(double lado){
   this->perimetro=lado * 4;
   return;
};

void Cuadrado::operator~(){
```

```
   cout << this->nombre << endl;
   cout << this->color << endl;
   cout << this->lado << endl;
};

void Cuadrado::operator!(){
    cout << this->area << endl;
    cout << this->perimetro << endl;
};


//Triangulo

Triangulo::Triangulo() {}

Triangulo::Triangulo(string nombre, string color, double base, double altura, double
    hipotenusa) {
        this->nombre=nombre;
        this->color=color;
   this->base=base;
   this->altura=altura;
   this->hipotenusa=hipotenusa;
        this->area;
        this->perimetro;
}

Triangulo::Triangulo(const Triangulo& orig) {};

Triangulo::~Triangulo() {};

void Triangulo::calcularArea(double base, double altura) {
    this->area=(base * altura / 2);
    return;
}

void Triangulo::calcularPerimetro(double base, double altura, double hipotenusa){
    this->perimetro=(base + altura + hipotenusa);
    return;
}

void Triangulo::operator~(){
   cout << this->nombre << endl;
   cout << this->color << endl;
   cout << this->base << endl;
   cout << this->altura << endl;
   cout << this->hipotenusa << endl;
}

void Triangulo::operator!(){
    cout << this->area << endl;
    cout << this->perimetro << endl;
}

//Circulo
```

```
Circulo::Circulo() {}

Circulo::Circulo(string nombre, string color, double radio) {
       this->nombre=nombre;
       this->color=color;
   this -> radio=radio;
       this->area;
       this->perimetro;
}

Circulo::Circulo(const Circulo& orig) {}

Circulo::~Circulo() {

}

void Circulo::calcularArea(double radio) {
    this->area=(radio * radio * PI);
    return;
}

void Circulo::calcularPerimetro(double radio){
    this->perimetro=(2 * PI * radio);
    return;
}

void Circulo::operator~(){
   cout << this->nombre << endl;
   cout << this->color << endl;
   cout << this->radio << endl;
}

void Circulo::operator!(){
    cout << this->area << endl;
    cout << this->perimetro << endl;
}
```

2.3. Archivo main.cpp

Archivo principal del programa en donde se ejecutan las creaciones de diferentes objetos y se llama a realizar los diferentes cálculos dentro de estas clases derivadas.

```
// main.cpp
#include "Figuras.h"
int main(int argc, char** argv) {



    Figuras p;
    p.nombre = "Figura\n";
    p.color = "Negro";
   p.calcularArea();
```

```
   p.calcularPerimetro();
   ~p;
        !p;

   Cuadrado q;
   q.nombre = "Cuadrado";
   q.color = "Blanco";
   q.lado = 3;
   q.calcularArea(3);
   q.calcularPerimetro(3);
        ~q;
        !q;

   Triangulo r;
   r.nombre = "Triangulo";
   r.color = "Azul";
   r.base = 3;
   r.altura = 3;
   r.hipotenusa = 3;
   r.calcularArea(3,3);
   r.calcularPerimetro(3,3,3);
   ~r;
        !r;

   Circulo s;
   s.nombre = "Circulo";
   s.color = "Rojo";
   s.radio = 3;
   s.calcularArea(3);
   s.calcularPerimetro(3);
   ~s;
        !s;

    return 0;
}
```

2.4. Sobrecarga de operadores

Para la sobrecarga de operadores se utilizó el siguiente fragmento de código:

```
// extracto de Figuras.cpp
void Figuras::operator~(){
   cout << this->nombre << this->color << endl;
};

void Figuras::operator!(){
    cout << "0" << endl;
    cout << "0" << endl;
};

void Cuadrado::operator~(){
   cout << this->nombre << this->color << this->lado << endl;
};
```

```
void Cuadrado::operator!(){
    cout << this->area << endl;
    cout << this->perimetro << endl;
};

void Triangulo::operator~(){
   cout << this->nombre << this->color << this->base << this->altura << this->hipotenusa
       << endl;
}

void Triangulo::operator!(){
    cout << this->area << endl;
    cout << this->perimetro << endl;
}

void Circulo::operator~(){
   cout << this->nombre << this->color << this->radio << endl;
}

void Circulo::operator!(){
    cout << this->area << endl;
    cout << this->perimetro << endl;
}
```

2.5. Archivo Makefile

Este archivo se genera automáticamente gracias a la herramienta de ambiente de programación NetBeans.

```
// Makefile
MKDIR=mkdir
CP=cp
CCADMIN=CCadmin

build: .build-post

.build-pre:

.build-post: .build-impl

clean: .clean-post

.clean-pre:

.clean-post: .clean-impl

clobber: .clobber-post

.clobber-pre:

.clobber-post: .clobber-impl
```

```
all: .all-post

.all-pre:

.all-post: .all-impl

build-tests: .build-tests-post

.build-tests-pre:

.build-tests-post: .build-tests-impl

test: .test-post

.test-pre: build-tests

.test-post: .test-impl

help: .help-post

.help-pre:

.help-post: .help-impl

include nbproject/Makefile-impl.mk

include nbproject/Makefile-variables.mk
```

3. Diagrama UML

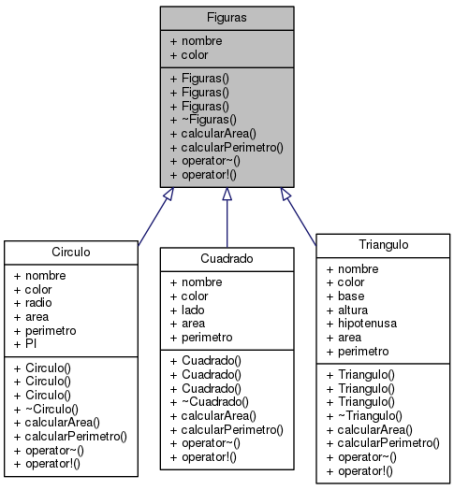

Figura 1: Se muestra el diagrama de jerarquía UML del programa.

4. Conclusiones

- Se termina el programa para calcular el área y el perímetro de los objetos de clases derivadas de la clase Figuras.
- Se logra sobrecargar los operadores indicados para realizar las distintas operaciones especificadas de impresión.
- Se utilizan los métodos virtuales dentro de las clases para realizar los cálculos indicados.