Reporte de laboratorio 2

Laura Rincón Riveros - B55863
Esteban Vargas Vargas - B16998
Grupo 3

11 de septiembre de 2016

Índice

---

1. Introducción

En el presente laboratorio se implentó la programación orientada a objetos, en conjunto con la herencia, poliformismo y sobrecarga de operadores. Como se presenta a continuación:

2. Desarrollo

2.1. Clase base

Se creó una clase base llamada Figura, en la cuál se definieron dos atributos y dos funciones virtuales, como se muestra en la Figura 1

```h
#ifndef FIGURA_H
#define FIGURA_H

#include <iostream>
using namespace std;

///Clase madre que le hereda a Circulo, Triangulo y Cuadrado
class Figura{
public:
  Figura();
  Figura(string nombre, string color);

  virtual ~Figura();

  string nombre;
  string color;

  virtual double area();
  virtual double perimetro();
};

#endif /* FIGURA_H */
```

(a) Figura.h

```cpp
#include "Figura.h"

///Constructor vacio de Figura
Figura::Figura(){

}
///Constructor que recibe los atributos nombre y color
Figura::Figura(string nombre, string color){
  this->nombre = nombre;
  this->color = color;
}

///Destructor de la clase figura
Figura::~Figura(){

}

///Función que devuelve el valor cero como área de una figura (es reimplementada por cada clase derivada)
double Figura::area(){
  return 0.0;
}
///Función que devuelve el valor cero como perímetro de una figura (es reimplementada por cada clase derivada)
double Figura::perimetro(){
  return 0.0;
}
```

(b) Figura.cpp

Figura 1: Clase base Figura

1. Figura.h : se declara el constructor, destructor, los atributos y las funciones que contiene la clase Figura.

    - Atributos: dos *string* nombre y color que puede o no recibir el constructor.

2. Figura.cpp : En este archivo se implementan las funciones definidas en Figura.h, por lo que se tiene que incluir en este archivo.

    - Área: Retorna un double. Se implementa de nuevo en las clases derivadas, por lo que se define como virtual.
    - Perímetro: Retorna un double. Se implementa de nuevo en las clases derivadas, por lo que se define como virtual.

2.2. Clases derivadas

Las clases derivadas son clases en las cuales se utilizó herencia con la clase madre Figura; por lo mismo, se tuvo que incluir el archivo "Figura.h". Sin embargo, cada clase cuenta con ciertos atributos propios de la misma y funciones para acoplar el objetivo a la figura específica.

Se crearon tres clases derivadas, como se presenta a continuación:

2.2.1. Triángulo

```h
#include "Figura.h"

///Clase Triangulo que hereda de la clase Figura
class Triangulo:public Figura{
public:
  Triangulo();
  Triangulo(double base, double altura);

  virtual ~Triangulo();

  double base;
  double altura;

  double area();
  double perimetro();
  void operator~();
  void operator!();
};
```

(a) Triangulo.h

```cpp
#include "Triangulo.h"

///Constructor vacío de la clase Triangulo
Triangulo::Triangulo(){

}

///Constructor de la clase Triangulo que recibe los atributos propios de la clase:base y altura
Triangulo::Triangulo(double base, double altura){
  this->base = base;
  this->altura = altura;
  this->color = "verde";
  this->nombre ="Dexter";
}

///Destructor de la clase Triangulo
Triangulo::~Triangulo(){

}

///Función que devuelve el valor del área de un triángulo
double Triangulo::area(){
  return (base*altura/2);
}

///Función que devuelve el valor del perimetro de un triángulo
double Triangulo::perimetro(){
  return (3*base);
}
```

(b) Triangulo.cpp

Figura 2: Clase Triangulo derivada de Figura

En la línea 4 del archivo Triangulo.h (Figura 2) se observa la implementación de la herencia. Además, como se aprecia en la Figura 2 hay dos atributos adicionales para el Triángulo, el valor de su base y altura. En el archivo Triangulo.cpp es donde se aprecia la implementación de las funciones de área y perímetro.

1. Atributos: dos *double*, base y altura.
2. Funciones:
   - Área: se utiliza la fórmula $A_{triangulo} = \frac{base*altura}{2}$, propia del área del tríangulo.
   - Perímetro: se utiliza la fórmula $P_{triangulo} = 3 * base$, propia del perímetro del tríangulo equilátero (Se generalizó la fórmula por comodidad).

2.2.2. Cuadrado

```
1  #include "Figura.h"
2
3  ///Clase Cuadrado que hereda de la clase Figura
4  class Cuadrado:public Figura{
5  public:
6    Cuadrado();
7    Cuadrado(double lado);
8
9    virtual ~Cuadrado();
10
11   double lado;
12
13   double area();
14   double perimetro();
15   void operator~();
16   void operator!();
17 };
```

(a) Cuadrado.h

```
1  #include "Cuadrado.h"
2
3  ///Constructor vacio de la clase Cuadrado
4  Cuadrado::Cuadrado(){
5
6  }
7
8  ///Constructor de la clase Cuadrado que recibe los atributos propios de la clase:lado
9  Cuadrado::Cuadrado(double lado){
10   this->lado = lado;
11   this->color = "azul";
12   this->nombre ="Sergio";
13 }
14
15 ///Destructor de la clase Cuadrado
16 Cuadrado::~Cuadrado(){
17
18 }
19
20 ///Función que devuelve el valor del área de un cuadrado
21 double Cuadrado::area(){
22   return (lado*lado);
23 }
24
25 ///Función que devuelve el valor del perímetro de un cuadrado
26 double Cuadrado::perimetro(){
27   return (4*lado);
28 }
```

(b) Cuadrado.cpp

Figura 3: Clase Cuadrado derivada de Figura

En la línea 4 del archivo Cuadrado.h (Figura 3) se observa la implementación de herencia.Se puede apreciar en la Figura que hay un atributo adicional para el Cuadrado, el valor de su lado.En el archivo Cuadrado.cpp es donde se aprecia la implementación de las funciones de área y perímetro.

1. Atributos: un *double* lado.
2. Funciones:
   - Área: se utiliza la fórmula $A_{cuadrado} = lado^2$, propia del área de un cuadrado.
   - Perímetro: se utiliza la fórmula $P_{cuadrado} = 4 * lado$, propia del perímetro de un cuadrado.

2.2.3. Círculo

```
1  #include "Figura.h"
2
3  ///Clase Circulo que hereda de la clase Figura
4  class Circulo:public Figura{
5  public:
6    Circulo();
7    Circulo(double radio);
8
9    virtual ~Circulo();
10
11   double radio;
12
13   double area();
14   double perimetro();
15   void operator~();
16   void operator!();
17 };
```

(a) Circulo.h

```
1  #include "Circulo.h"
2
3  ///Constructor vacio de la clase Circulo
4  Circulo::Circulo(){
5
6  }
7
8  ///Constructor de la clase Circulo que recibe los atributos propios de la clase:radio
9  Circulo::Circulo(double radio){
10   this->radio = radio;
11   this->color = "rojo";
12   this->nombre ="Beto";
13 }
14
15 ///Destructor de la clase Circulo
16 Circulo::~Circulo(){
17
18 }
19
20 ///Función que devuelve el valor del área de un círculo
21 double Circulo::area(){
22   return (3.141592*radio*radio);
23 }
24
25 ///Función que devuelve el valor del perímetro de un círculo
26 double Circulo::perimetro(){
27   return (2*3.141592*radio);
28 }
```

(b) Circulo.cpp

Figura 4: Clase Circulo derivada de Figura

En la línea 4 del archivo Circulo.h (Figura 4) se observa la implementación de herencia. Se puede apreciar en la Figura que hay un atributo adicional para el Círculo, el valor de su radio. En el archivo Circulo.cpp, es donde se aprecia la implementación de las funciones de área y perímetro.

1. Atributos: un *double* radio.
2. Funciones:
   - Área: se utiliza la fórmula $A_{circulo} = \pi * r^2$, propia del área del círculo.
   - Perímetro: se utiliza la fórmula $P_{circulo} = 2 * \pi * r$, propia del perímetro del círculo.

2.3. Sobrecarga de operadores

La sobrecarga de operadores se utilizó con los operadores ! y ∼ para que, al llamar al primero, se imprimieran los datos de cada figura y al llamar al segundo se imprimieran los valores de área y perímetro respectivos. A continuación se presenta la implementación mencionada:

```
31 ///Sobrecarga del operador ~
32 void Triangulo::operator~(){
33   cout << "El nombre del triángulo es: " << nombre << endl;
34   cout << "El color del triángulo es: " << color << endl;
35   cout << "La base del triángulo es: " << base << endl;
36   cout << "La altura del triángulo es: " << altura << endl;
37 }
38
39 ///Sobrecarga del operador !
40 void Triangulo::operator!(){
41   cout << "El área del triángulo es: " << area() << endl;
42   cout << "El perímetro del triángulo es: " << perimetro() << endl;
43 }
```

(a) Tríangulo.cpp

```
30 ///Sobrecarga del operador ~
31 void Cuadrado::operator~(){
32   cout << "El nombre del cuadrado es: " << nombre << endl;
33   cout << "El color del cuadrado es: " << color << endl;
34   cout << "El lado del cuadrado es: " << lado << endl;
35 }
36
37 ///Sobrecarga del operador !
38 void Cuadrado::operator!(){
39   cout << "El área del cuadrado es: " << area() << endl;
40   cout << "El perímetro del cuadrado es: " << perimetro() << endl;
41 }
```

(b) Circulo.cpp

```
30 ///Sobrecarga del operador ~
31 void Circulo::operator~(){
32   cout << "El nombre del circulo es: " << nombre << endl;
33   cout << "El color del circulo es: " << color << endl;
34   cout << "El radio del circulo es: " << radio << endl;
35 }
36
37 ///Sobrecarga del operador !
38 void Circulo::operator!(){
39   cout << "El área del círculo es: " << area() << endl;
40   cout << "El perímetro del círculo es: " << perimetro() << endl;
41 }
```

(c) Circulo.h

Figura 5: Sobrecarga de operadores

Para realizar la sobrecarga de los operadores, también hay que declarar dichas funciones (tipo void) en los headers, esto se muestro en las figuras 2, 3 y 4 en las líneas 15-17.

2.4. Main

En el archivo main.cpp se incluyeron todos los archivos headers con el fin de poder hacer uso de los objetos respectivos. Se creó un objeto de cada clase derivada, estableciendo sus atributos para luego calcular área y perímetro de éstos. Además, se le aplicaron los operadores sobrecargados a los objetos Triangulo, Circulo y Cuadrado.

```cpp
#include <cstdlib>
#include "Figura.h"
#include "Triangulo.h"
#include "Circulo.h"
#include "Cuadrado.h"
using namespace std;

int main (int argc, char** argv){

  Triangulo* t= new Triangulo(6.0,5.0);
  ~(*t);
  !(*t);
  //double at = t->area(); ERA PARA COMPROBAR VALORES
  //double pt = t->perimetro(); ERA PARA COMPROBAR VALORES
  delete t;

  Circulo* c= new Circulo(6.0);
  ~(*c);
  !(*c);
  //double ac = c->area(); ERA PARA COMPROBAR VALORES
  //double pc = c->perimetro(); ERA PARA COMPROBAR VALORES
  delete c;

  Cuadrado* cu= new Cuadrado(7.0);
  ~(*cu);
  !(*cu);
  //double acu = cu->area(); ERA PARA COMPROBAR VALORES
  //double pcu = cu->perimetro(); ERA PARA COMPROBAR VALORES
  delete cu;

}
```

Figura 6: Contenido de main.cpp

Y su ejecución:

```
laura@macondo:~/Downloads/IE-0217-II-16-Lab2/src$ make
g++ -Wall -o main *.cpp
echo "Ejecución del programa"
Ejecución del programa
./main
El nombre del triángulo es: Dexter
El color del triángulo es: verde
La base del triángulo es: 6
La altura del triángulo es: 5
El área del triángulo es: 15
El perímetro del triángulo es: 18
El nombre del círculo es: Beto
El color del círculo es: rojo
El radio del círculo es: 6
El área del círculo es: 113.097
El perímetro del círculo es: 37.6991
El nombre del cuadrado es: Sergio
El color del cuadrado es: azul
El lado del cuadrado es: 7
El área del cuadrado es: 49
El perímetro del cuadrado es: 28
rm main
```

Figura 7: ejecución del main

2.5. Diagrama de clases

Para una mejor visualización de la composición de la clase base y sus derivadas, se presenta a continuación un diagrama de clases:

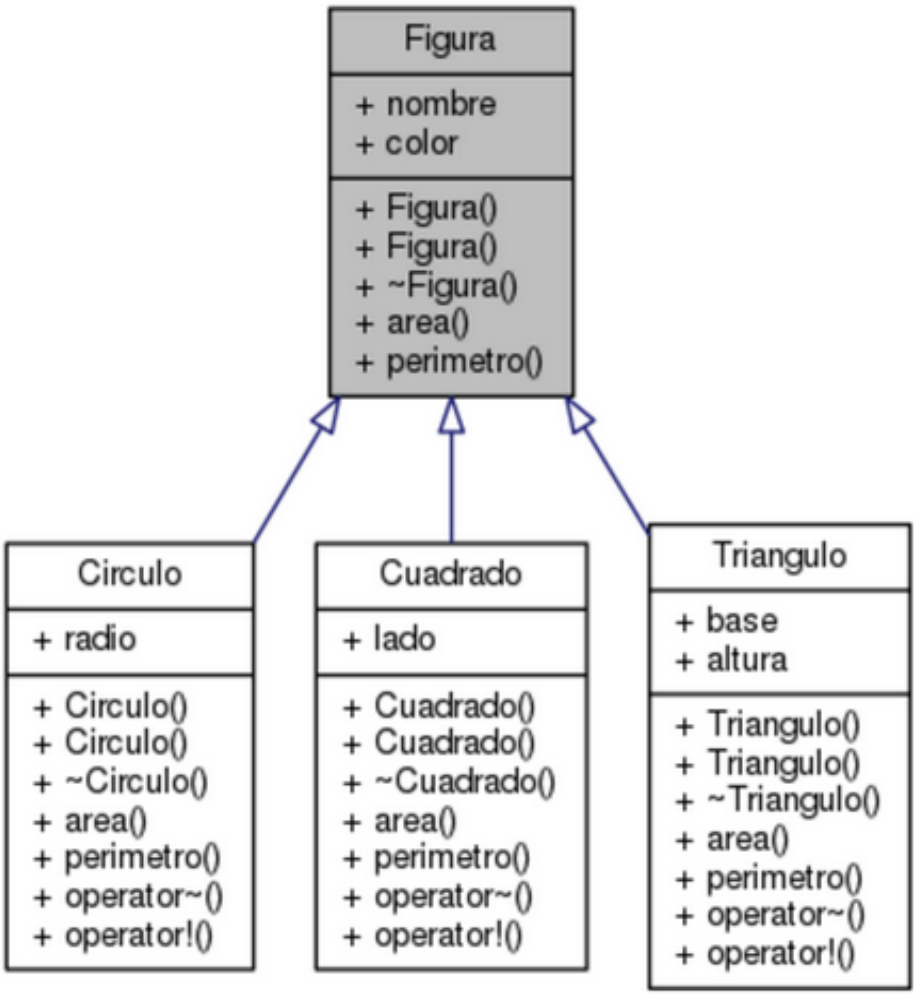

Figura 8: Diagrama de clases

3. Conclusiones

- Se empleó la Programación Orientada a Objetos, para el uso de clases.
- Se logró heredar atributos y características de la clase base a sus clases derivadas.
- Se implementó el poliformismo para que cada clase reimplementara funciones según sus requerimientos.
- Se sobrecargaron los operadores ! y $\sim$ para que cumplieran una función determinada al usarse con objetos.