4 Composición en C++

Elaborado por: Ukranio Coronilla

Es posible que un objeto sea miembro de una clase, a esto se le conoce como composición. Un ejemplo de composición se muestra en el programa 4-1, donde la clase `Coordenada` permite almacenar una coordenada del plano cartesiano, y es miembro de la clase `Rectangulo`. Compile y ejecute el código para probar su correcto funcionamiento.

Programa 4-1

```
1  #include <iostream>
2  using namespace std;
3
4  class Coordenada
5  {
6  private:
7     double x;
8     double y;
9  public:
10    Coordenada(double = 0, double = 0);
11    double obtenerX();
12    double obtenerY();
13 };
14
15 class Rectangulo
16 {
17 private:
18    Coordenada superiorIzq;
19    Coordenada inferiorDer;
20 public:
21    Rectangulo();
22    Rectangulo(double xSupIzq, double ySupIzq, double xInfDer, double yInfDer);
23    void imprimeEsq();
24    Coordenada obtieneSupIzq();
25    Coordenada obtieneInfDer();
26 };
27
28 int main( )
29 {
30    Rectangulo rectangulo1(2,3,5,1);
31    double ancho, alto;
32
34    cout << "Calculando el área de un rectángulo dadas sus coordenadas en un plano
cartesiano:\n";
35    rectangulo1.imprimeEsq();
36
37    alto = rectangulo1.obtieneSupIzq().obtenerY() -
rectangulo1.obtieneInfDer().obtenerY();
38    ancho = rectangulo1.obtieneInfDer().obtenerX() -
rectangulo1.obtieneSupIzq().obtenerX();
39    cout << "El área del rectángulo es = " << ancho*alto << endl;
40    return 0;
41 }
42
43 Coordenada::Coordenada(double xx, double yy) : x(xx), y(yy)
44 { }
```

```
45
46 double Coordenada::obtenerX()
47 {
48    return x;
49 }
50
51 double Coordenada::obtenerY()
52 {
53    return y;
54 }
55
56 Rectangulo::Rectangulo() : superiorIzq(0,0), inferiorDer(0,0)
57 { }
58
59 Rectangulo::Rectangulo(double xSupIzq, double ySupIzq, double xInfDer, double
yInfDer):superiorIzq(xSupIzq, ySupIzq), inferiorDer(xInfDer, yInfDer)
60 { }
61
62 void Rectangulo::imprimeEsq()
63 {
64    cout << "Para la esquina superior izquierda.\n";
65    cout << "x = " << superiorIzq.obtenerX() << " y = " << superiorIzq.obtenerY() << endl;
66    cout << "Para la esquina inferior derecha.\n";
67    cout << "x = " << inferiorDer.obtenerX() << " y = " << inferiorDer.obtenerY() << endl;
68 }
69
70 Coordenada Rectangulo::obtieneSupIzq()
71 {
72    return superiorIzq;
73 }
74
75 Coordenada Rectangulo::obtieneInfDer()
76 {
77    return inferiorDer;
78 }
```

En este código se puede notar que la inclusión de una clase dentro de otra no tiene nada de especial. La parte novedosa tiene que ver con la forma en que se invoca al constructor de la clase miembro, dentro del constructor de la clase contenedora. Esto se puede observar en la línea 59, cuando con `superiorIzq(xSupIzq, ySupIzq)` se invoca al constructor de la clase `Coordenada`. Observe también en la línea 37 como se mandan a llamar métodos en una clase que pertenece a otra.

Ejercicio 1: En este caso tenemos dos clases y un programa principal, separe el código en cinco archivos, dos de implementación, dos de interfaz y un programa principal. Posteriormente para compilar utilice un archivo Makefile y el programa make. Suponiendo que nombra los archivos como Rectangulo.cpp Rectangulo.h Coordenada.cpp Coordenada.h y prac_compo.cpp. El archivo Makefile quedaría como sigue (es importante que en las líneas de compilación el primer espacio se un tabulador):

```
prac_compo: prac_compo.cpp Rectangulo.o
        g++ prac_compo.cpp Rectangulo.o Coordenada.o -o prac_compo
Rectangulo.o: Rectangulo.cpp Rectangulo.h
        g++ Rectangulo.cpp -c
Coordenada.o: Coordenada.cpp Coordenada.h
        g++ Coordenada.cpp -c
```

Ejercicio 2: Para hacer más legible el programa y reutilizar código, modifique la línea 30 para que se pueda crear el objeto `Rectangulo` con la siguiente línea de código:

```
Rectangulo rectangulo1(Coordenada(2,3),Coordenada(5,1));
```

Ejercicio 3: Implemente el método `obtieneArea` dentro de la clase `Rectangulo` por ser más conveniente al ocultar los métodos de la clase `Coordenada`, al usuario de la clase `Rectangulo`.

"Dado que la herencia es tan importante en la POO, casi siempre se enfatiza mucho su uso, de manera que los programadores novatos pueden llegar a pensar que hay que emplearla en todas partes. Esto puede dar lugar a que se hagan diseños demasiados complejos y complicados. En lugar de esto, en primer lugar, cuando se van a crear nuevas clases debe considerarse la composición, ya que es más simple y **flexible**. "

Bruce Eckel - "Thinking in Java"

Al usar la composición si se realizan cambios en la clase `Coordenada`, es posible cambiar el comportamiento de la clase `Rectangulo` y no ser necesaria su modificación. Esta es la flexibilidad que proporciona la composición.

Ejercicio 4: Pruebe la flexibilidad modificando la clase `Coordenada` para que reciba los dos parámetros como coordenadas polares (r, Θ) sin modificar la clase `Rectangulo`. Asigne los valores apropiados para que devuelva la misma área que el ejercicio anterior.

Ejercicio 5: Utilice composición para elaborar una clase `Ortoedro` la cual se compone de 6 objetos `Rectangulo`, y donde cada objeto `Coordenada` es de tres dimensiones. Se debe disponer de los metodos: `obtieneVertices`, `obtieneArea` y `obtieneVolumen` para el área de la superficie y el volumen. La inicialización del objeto se debe dar mediante las 2 coordenadas de los vértices opuestos donde el primer vértice está cerca del origen y el segundo lejos, dentro ambos del primer octante. Divida la programación entre los miembros del equipo para terminar a tiempo.

Proyecto 1 para subir al MOODLE (continúa):

6.- Repita el ejercicio 5 pero en lugar de una clase Ortoedro, hacer la clase `Paralelepipedo` donde un vértice se encuentra en el origen de coordenadas, tal como muestra la figura 4-1. Recuerde que áreas y volúmenes en este caso se pueden encontrar con ayuda del producto cruz y del producto punto. Por esta razón en lugar de la clase `Coordenada`, utilice la clase `Vector`. En el archivo a subir en MOODLE utilice una línea con carácter "-" para la separación entre la interfaz, implementación y código de un mismo proyecto.

$|[\vec{u},\vec{v},\vec{w}]|$ = Volumen del PARALELEPÍPEDO

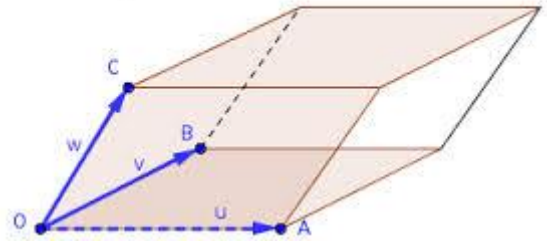

Figura 4-1