8 Herencia

Elaborado por: Ukranio Coronilla

Herencia es el proceso mediante el cual se crea una nueva clase denominada clase derivada, a partir de una clase existente llamada clase base. De manera automática la clase derivada tendrá las mismas variables y funciones miembro de la clase base. Un ejemplo de su uso se muestra en el programa 8-1, este programa crea una clase derivada `Cilindro`, a partir de la clase base `Circulo`. Compile y ejecute el código para probar su funcionamiento.

Programa 8-1

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  const double PI = acos(-1.0);
6
7  class Circulo
8  {
9  private:
10    double radio;
11 public:
12    Circulo(double = 1.0);
13    double calcval();
14 };
15
16 Circulo::Circulo(double r) : radio(r)
17 { }
18
19 double Circulo::calcval() //Calcula el área del círculo
20 {
21    return(PI * radio * radio);
22 }
23
24 class Cilindro : public Circulo
25 {
26 private:
27    double longitud;
28 public:
29    Cilindro(double r = 1.0, double l = 1.0) : Circulo(r), longitud(l) {}
30    double calcval();
31 };
32
33 double Cilindro::calcval()  //Calcula el volumen del cilindro
34 {
35   return (longitud * Circulo::calcval());
36 }
37
38 int main()
39 {
40   Circulo Circulo_1, Circulo_2(2);
41   Cilindro Cilindro_1(3,4);
42
43   cout << "El área de Circulo_1 es " << Circulo_1.calcval() << endl;
44   cout << "El área de Circulo_2 es " << Circulo_2.calcval() << endl;
45   cout << "El volumen de Cilindro_1 es " << Cilindro_1.calcval() << endl;
```

```
46   Circulo_1 = Cilindro_1;
47   cout << "\nEl área de Circulo_1 es ahora " << Circulo_1.calcval() << endl;
48   return 0;
49 }
```

En la línea 2 se incluye la librería que permite el uso de funciones matemáticas como el arco coseno (`acos`) de la línea 5. Se utiliza la función arco coseno para obtener el valor de pi con la mayor exactitud que la computadora puede permitir.

Ejercicio 1: Sabiendo que

π = 3.14159 26535 89793 23846 26433 83279 50288 41971 69399 37510…

Imprima `PI` con 40 cifras de la parte fraccionaria, ¿Para un dato tipo `double`, hasta cuantas cifras decimales de precisión alcanza?, ¿Y para `float`? ¿Y para `long double`? ¿Cuántos bytes ocupa cada una de estas variables en memoria?

En la línea 7 se declara la clase `Circulo` cuya función `calcval` se encarga de calcular el área de un circulo, y cuyo radio se establece a 1.0 por omisión como lo indica la línea 12.

La línea 24 muestra cómo se crea la clase derivada `Cilindro` a partir de la clase base `Circulo`. La clase derivada entonces tendrá todos los miembros de la clase base, y también el dato miembro `longitud`, así como un constructor que en este caso se especifica por completo en la línea 29. Observe la implementación que se hace de manera similar a la línea 16 pero dentro de la declaración de la clase.

Con premeditación se ha dejado el mismo nombre de la función miembro (`calcval`) a ambas clases, lo cual no provoca ningún conflicto pues se trata de clases distintas, aunque una se derive de otra.

En la línea 46 se utiliza el hecho de que **un objeto de la clase derivada puede asignarse a un objeto de la clase base**. Esto es posible debido a que ambas clases tienen los mismos datos miembros de la clase base, y son precisamente estos miembros los que se asignan, como se puede comprobar en la impresión de la línea 47.

Ejercicio 2: Pruebe si es posible que un objeto de la clase base pueda asignarse a un objeto de la clase derivada. Pruebe también si es posible en la línea 35 calcular el área del círculo haciendo uso de la variable miembro `radio`.

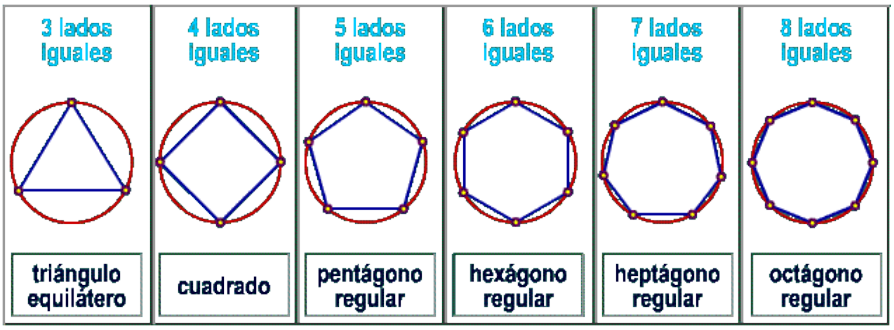
Como pudo observar en el ejercicio anterior, no es posible acceder a una variable miembro privada de la clase base desde la implementación de la clase derivada. Para que una clase derivada pueda acceder a los miembros de una clase base, pero que aun así, estos sigan estando protegidos

para otras clases, es necesario substituir `private` por `protected`. De esta manera los miembros protegidos podrán ser accedidos por su nombre en cualquiera de las clases descendentes, no solo en la clase derivada.

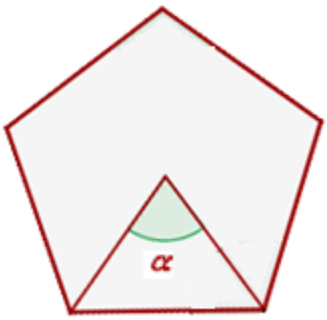
Ejercicio 3: Pruebe la aseveración anterior con una clase descendente de Cilindro, llamada toroide.

En general la relación entre las clases en la herencia cumplen con una relación "es un", mientras que en la composición se cumple una relación "tiene un". Si reflexiona un poco en el mundo real, es más común la relación de composición que la de herencia. De hecho para el programa 8-1 está mejor dicho que un `Cilindro` "tiene un" `Circulo` a decir que un `Cilindro` "es un" `Circulo` ¿no cree?

Ejercicio 4: Vamos a elaborar un programa para calcular numéricamente el valor de π con distinta precisión. Primero, reutilizando el ejercicio resuelto 2 del tema 5, vamos a derivar una clase que representa a los polígonos regulares, inscritos en una circunferencia de radio uno llamada `PoligonoReg` a partir de la clase `PoligonoIrreg`.



En este caso usamos herencia porque podemos afirmar, que un polígono regular es un subconjunto de los polígonos irregulares (sin embargo lo contrario no es verdad). La clase `PoligonoReg` solo tendrá como datos miembro el número de vértices del polígono regular, y el ángulo α que se subtiende en cada triangulo que lo compone como muestra la figura:



El constructor debe recibir como único parámetro el número de vértices, y el único método de la clase será `obtieneArea`, el cual devuelve el área del polígono regular.

En la función principal se deberán imprimir las coordenadas de cada vértice, así como las áreas obtenidas para polígonos de 3, 4, 5, 6… lados.

¿Si se utiliza el tipo de dato `float` en todas las variables del programa, cual es la máxima precisión (de la parte fraccionaria) con la que su programa podría aproximarse a π? Pruébelo.

¿Cuál sería la máxima precisión con la que su programa puede calcular π usando `double`?