Qué es el polimorfismo?

En POO, el *polimorfismo* permite que diferentes objetos respondan de modo diferente al mismo mensaje.

El polimorfismo adquiere su máxima potencia cuando se utiliza en unión de herencia.

Por ejemplo, si figura es una clase base de la que cada figura geométrica hereda características

comunes, C++ permite que cada clase utilice una función (método) Copiar como nombre de una función

miembro.

```
class figura {
   tipoenum tenum;          //tipoenum es un tipo enumerado


public:
   virtual void Copiar();
   virtual void Dibujar();
   virtual double Area);
};

class circulo : public figura {
   ...
public:
   void Copiar();
   void Dibujar();
   double Area();
};

clas rectangulo : public figura {
   ...
public:
   void Copiar(); // el polimorfismo permite que objetos diferentes
                  // tengan idénticos nombres de funciones miembro
   void Dibujar();
   double Area();
};
```

Otro ejemplo se puede apreciar en la jerarquía de clases:

```
class Poligono {      // superclase
public:
   float Perimetro();
   virtual float Area();
   virtual boolean PuntoInterior();
protected:
   void Visualizar();
};

class Rectangulo : public Poligono{

public:
   virtual float Area();
   virtual boolean PuntoInterior();
   void fijarRectangulo();
private:
   float Alto;
   float Bajo;
   float Izquierdo;
   float Derecho;
};
```

14.8. USO DEL POLIMORFISMO

El polimorfismo permite utilizar el mismo interfaz, tal como métodos (funciones miembro) denominados `dibujar` y `calcular_area`, para trabajar con toda clase de figuras.

La forma más adecuada de usar polimorfismo es a través de punteros. Supongamos que se dispone de una colección de objetos `figura` en una estructura de datos, tal como un array o una lista enlazada. El array almacena simplemente punteros a objetos `figura`. Estos punteros apuntan a cualquier tipo de figura. Cuando se actúa sobre estas figuras, basta simplemente recorrer el array con un bucle e invocar a la función miembro apropiada mediante el puntero a la instancia. Naturalmente, para realizar esta tarea las funciones miembro deben ser declaradas como virtuales en la clase `figura`, que es la clase base de todas las figuras geométricas.

Para poder utilizar polimorfismo en C++ se deben seguir las siguientes reglas:

1. Crear una jerarquía de clases con las operaciones importantes definidas por las funciones miembro declaradas como virtuales en la clase base.
2. Las implementaciones específicas de las funciones virtuales se deben hacer en las clases derivadas. Cada clase derivada puede tener su propia versión de las funciones. Por ejemplo, la implementación de la función `dibujar` varía de una figura a otra.
3. Las instancias de estas clases se manipulan a través de una referencia o un puntero. Este mecanismo es la ligadura dinámica base.

Se obtiene ligadura dinámica sólo cuando las funciones miembro virtuales se invocan a través de un puntero a una instancia de una clase base.

Importancia del polimorfismo en la programación orientada a objetos:

- Flexibilidad en el diseño de software.
- Reutilización de código.
- Facilidad de mantenimiento.
- Abstracción de conceptos.

override es una palabra reservada que está disponible desde el estándar C++11 y

únicamente sirve para que el compilador nos ayude a detectar inconsistencias al sobreescribir funciones. Cuando tu marcas una función como override lo único que haces es indicarle al compilador que debe verificar lo siguiente:

- Que en alguna clase padre existe una función con **exactamente** la misma declaración
- PPQue la función de la clase padre está marcada como virtual

Programación C++ Segunda edición

Algoritmos, estructuras de datos y objetos – Luis Joyanes

14.7 Polimorfismo pag 527