Programación Orientada a Objetos

Tarea 4.2. Polimorfismo

José Fidel Argudo Argudo Francisco Palomo Lozano
Inmaculada Medina Bulo Gerardo Aburruzaga García
Pedro Delgado Pérez

Versión 2.0

Tarea 4.2. Cuestiones

Ejercicio 1

Sea cierta clase base `B` y una derivada `D`. Ambas tienen cierto método `f()`, pero en `B` se define como virtual puro.

1. Escriba la definición de `B::f()` (no recibe parámetros ni devuelve nada).
2. La clase `B` es una clase a la que se le denomina ..., ¿cómo?
3. ¿Qué hace el fragmento de código siguiente?

```
1 B b, *bp;
2 D d;
3 bp = &d;
4 bp->f();
```

Tarea 4.2. Cuestiones

Ejercicio 2

Consideremos la siguiente jerarquía de clases:

```
1  struct V {
2     virtual void fv() = 0;
3     virtual ~V() {}
4  };
5  struct X : V {
6     void fv() {}
7  };
8  struct Y : V {
9     void fv() {}
10 };
11 struct Z : V {
12    void fv() {}
13 };
```

Tarea 4.2. Cuestiones

Ejercicio 2 (cont.)

Supongamos una función `f()` para procesar objetos de la clase `V`:

```
1  void f(V& v)
2  {
3     if (typeid(v) == typeid(X)) {
4        std::cout << "Procesando␣objeto␣X...\n";
5        // código específico para X
6     }
7     if (typeid(v) == typeid(Y)) {
8        std::cout << "Procesando␣objeto␣Y...\n";
9        // código específico para Y
10    }
11    if (typeid(v) == typeid(Z)) {
12       std::cout << "Procesando␣objeto␣Z...\n";
13       // código específico para Z
14    }
15 }
```

Tarea 4.2. Cuestiones

Ejercicio 2 (cont.)

1. ¿Existe una relación de realización entre las clases presentadas? ¿Por qué?
2. ¿Cuál es la salida del siguiente código?

```
X x; V* pv = new Y;
f(x); f(*pv);
```

3. ¿Es la mejor forma de implementar el comportamiento polimórfico de `f()`? Razone la respuesta. En caso negativo, describa cómo mejorar la implementación y, si es necesario, modifique el código anterior para que produzca la misma salida.

Tarea 4.2. Cuestiones

Ejercicio 3

1. Defina una clase paramétrica llamada `Buffer` para representar una zona de memoria, cuyos parámetros sean el tipo base de cada elemento de esa zona (por omisión, el tipo cuyo tamaño es 1 byte), y el tamaño de dicha zona (por omisión, 256 elementos). Defina dentro de la clase el atributo principal, que será un vector paramétrico (de la STL), y el constructor predeterminado.
2. A continuación defina un objeto de tipo `Buffer` formado por 128 elementos de tipo `int`, y otro formado por 256 elementos del tipo por omisión.

Tarea 4.2. Cuestiones

Ejercicio 4

1. Escriba la salida del siguiente programa.
2. ¿Qué ocurriría si las clases B<id> no fueran polimórficas?

Tarea 4.2. Cuestiones

```
1  #include <iostream>
2  #include <typeinfo>
3  using std::cout; using std::endl;

5  template<int id> class B {
6     int* p;
7  public:
8     B(): p{new int} {
9        cout << typeid(*this).name() << "::"
10            << typeid(*this).name() << "()" << endl;
11    }
12    B(const B& b): p{new int{*(b.p)}} {
13       cout << typeid(*this).name() << "::" << typeid(*this).name()
14            << "(const␣" << typeid(*this).name() << "&)" << endl;
15    }
16    virtual ~B() {
17       delete p;
18       cout << typeid(*this).name() << "::~"
19            << typeid(*this).name() << "()" << endl;
20    }
21 };
```

Tarea 4.2. Cuestiones

```
23 class D: public B<0> {
24 public:
25   D() { cout << "D::D()" << endl; }
26   D(const D& d): B<0>{static_cast<const B&>(d)}, b1{d.b1}, b2{d.b2}
27   { cout << "D::D(const␣D&)" << endl; }
28  ~D() { cout << "D::~D()" << endl; }
29 private:
30   B<1> b1;
31   B<2> b2;
32 };

34 int main() {
35    B<0>& b{*new D};
36    cout << "-----------------------" << endl;
37    D d{dynamic_cast<D&>(b)};
38    cout << "-----------------------" << endl;
39    delete &b;
40    cout << "-----------------------" << endl;
41 }
```