Programación Orientada a Objetos

Tarea 2.2. Constructores y uso de objetos

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

Versión 2.0

Tarea 2.2. Cuestiones

Ejercicio 1

Enumere las diferencias existentes entre inicializar un atributo en la lista de inicialización y asignarle un valor en el cuerpo del constructor.

Ejercicio 2

La lista de inicialización y la lista inicializadora, ¿cumplen la misma función en la construcción de objetos? ¿Es posible utilizar estas listas en otros métodos de una clase?

Tarea 2.2. Cuestiones

Ejercicio 3

```
1 class punto {
2   double x, y;
3 public:
4   punto(double a = 0., double b = 0.) : x{a}, y{b} {}
5   punto(const punto& p) : x{p.x}, y{p.y} {}
6   punto& operator =(const punto& p)
7     { x = p.x; y = p.y; return *this; }
8 };
```

Diga qué función de la clase `punto` se llama en cada una de las siguientes líneas. Si alguna depende de una línea anterior que sea incorrecta, corríjala previamente.

```
1 punto p;
2 punto q();
3 punto r(2.,);
4 punto s{3.4};
```

```
5 punto t{};
6 punto u(q);
7 punto v = r;
8 t = s;
```

Tarea 2.2. Cuestiones

Ejercicio 4

Sean las clases `Libro1` y `Libro2`:

```
1  class Libro1 {
2    string titulo_;
3    int pags_;
4  public:
5    Libro1(string t = "", int p = 0);
6    // ...
7  };
```

```
9  class Libro2 {
10   string titulo_;
11   int pags_;
12 public:
13   Libro2(string t, int p = 0);
14   Libro2(const char* c);
15   // ...
16 };
```

Decida si `X` se puede sustituir por `1`, `2` o ambos en los siguientes items:

1. Se puede definir: `LibroX lib1;`
2. Se tiene un constructor de conversión de `std::string` a `LibroX`
3. Se puede definir: `LibroX lib2[5];`

Tarea 2.2. Cuestiones

Ejercicio 4 (cont.)

```
1  class Libro1 {
2    string titulo_;
3    int pags_;
4  public:
5    Libro1(string t = "", int p = 0);
6    // ...
7  };
```

```
9  class Libro2 {
10   string titulo_;
11   int pags_;
12 public:
13   Libro2(string t, int p = 0);
14   Libro2(const char* c);
15   // ...
16 };
```

4. Se puede definir: `std::vector<LibroX> lib3;`
5. Siendo `"El␣Quijote"` una cadena literal de tipo `const char*`; se produce una conversión implícita a `string` al ejecutar:

   ```
   LibroX* lib4 = new LibroX("El␣Quijote");
   ```

6. Se puede definir: `LibroX lib5 = "El␣Quijote";`
7. Hace falta definir el destructor para `LibroX`.

Tarea 2.2. Cuestiones

Ejercicio 5

Considere la siguiente clase `Libro`:

```
2  #include <iostream>
3  #include <cstring>
4  using namespace std;

6  class Libro {
7    char* titulo_; int paginas_;
8  public:
9    Libro() : titulo_(new char[1]), paginas_(0) {*titulo_= 0;}
10   Libro(const char* t, int p) : paginas_(p) {
11     titulo_ = new char[strlen(t) + 1];
12     strcpy(titulo_, t);
13   }
14   ~Libro() { delete[] titulo_; }
15   int paginas() const { return paginas_; }
16   char* titulo() const { return titulo_; }
17 };
```

Tarea 2.2. Cuestiones

Ejercicio 5 (cont.)

Diga si el siguiente programa funciona correctamente. En caso afirmativo indique lo que imprime. En caso negativo haga las modificaciones necesarias para que funcione correctamente.

```
18 void mostrar(Libro l) {
19   cout << l.titulo() << "␣tiene␣"
20        << l.paginas() << "␣páginas" << endl;
21 }

23 int main() {
24   Libro l1("Fundamentos␣de␣C++", 474), l2;

26   l2 = l1;
27   mostrar(l2);
28 }
```

Tarea 2.2. Cuestiones

Ejercicio 6

```
1  struct B;          // Declaración adelantada
2  struct A {
3    A(B);            // Constructor de conversión de B a A
4  };
5  struct B {
6    operator A();    // Operador de conversión de B a A
7  };

9  void f(const A&); // Recibe por referencia un objeto constante de A

11 int main() {
12    B b; f(b);
13 }
```

1. Describa el error de compilación que provoca el código anterior. ¿Cómo modificaría las clases sin suprimir métodos para solucionarlo?
2. Suponga que el parámetro de `f()` es de entrada y salida y la línea 9 es sustituida por `void f(A&);`. ¿Qué error de compilación se produce? ¿Y cómo se puede resolver sin suprimir métodos?