CS2023 - Programación III

Fundamentos de Programación - UDF class

Rubén Rivas Medina

Marzo 2025

Objetivo

El propósito de este ejercicio es practicar el uso de constructores, destructores y, especialmente, la **Regla de los 3 y Regla de 5** mediante la implementación de una clase que gestione memoria dinámica.

1. Clases a implementar

Clase Libro

Implementar una clase sencilla que represente un libro. Sus miembros deben incluir:

- **Atributos privados**:
  - std::string titulo
  - std::string autor
  - int anio
- **Métodos públicos**:
  - Constructor que inicialice todos los atributos.
  - Método mostrarInformacion() que imprima los datos del libro.

Clase Biblioteca

Esta clase debe gestionar un arreglo dinámico de objetos Libro. Su implementación debe incluir:

- **Atributos privados**:
  - Libro* libros – arreglo dinámico de libros.
  - int cantidad – cantidad actual de libros.
  - int capacidad – capacidad máxima actual del arreglo.
- **Métodos públicos**:
  - Constructor por defecto: inicializa con capacidad mínima (por ejemplo, 2).
  - agregarLibro(const Libro&): agrega un libro y realoca memoria si es necesario.

- mostrarLibros() const: muestra la información de todos los libros.
- **Constructor de copia**: realiza una copia profunda del arreglo.
- **Operador de asignación**: realiza también una copia profunda, liberando memoria previa.
- **Destructor**: libera la memoria utilizada por el arreglo dinámico.

Regla de los 5 (Opcional)

Se recomienda también implementar:

- Constructor de movimiento.
- Operador de asignación por movimiento.

2. Recomendaciones

- Usar new y delete para gestionar memoria dinámica.
- Evitar memory leaks.
- Asegurarse de que el constructor de copia y el operador de asignación funcionen correctamente con múltiples objetos.

3. Ejemplos de uso

Ejemplo # 1

```
// Crear una biblioteca y agregar libros
Biblioteca biblio;
biblio.agregarLibro(Libro("1984", "George␣Orwell", 1949));
biblio.agregarLibro(Libro("Cien␣años␣de␣soledad", "Gabriel␣García␣Márquez", 1967));
biblio.mostrarLibros();

// Constructor de copia
Biblioteca copia = biblio;

// Operador de asignación (copia)
Biblioteca otra;
otra = biblio;

// Constructor por movimiento
Biblioteca movida = std::move(biblio);

// Intentar usar biblio después del move (puede causar crash si no está bien manejado)
std::cout << "biblio␣después␣del␣move" << std::endl;
biblio.mostrarLibros(); // <- Aquí podría fallar

// Operador de asignación por movimiento
Biblioteca destino;
destino = std::move(copia);

// Intentar usar copia después del move (potencial undefined behavior)
std::cout << "copia␣después␣del␣move" << std::endl;
copia.mostrarLibros(); // <- También puede fallar
```

Ejemplo # 2

```
// Crear una biblioteca y agregar libros
Biblioteca biblio;
biblio.agregarLibro(Libro("1984", "George Orwell", 1949));
biblio.agregarLibro(Libro("Cien años de soledad", "Gabriel García Márquez", 1967));
biblio.mostrarLibros();

// Constructor de copia
Biblioteca copia = biblio;

// Operador de asignación (copia)
Biblioteca otra;
otra = biblio;

// Constructor por movimiento
Biblioteca movida = std::move(biblio);

// Operador de asignación por movimiento
Biblioteca destino;
destino = std::move(copia);
```

4. Criterios de evaluación (opcional para clase)

- Uso correcto de la regla de los 3 (o 5).
- Manejo adecuado de memoria dinámica.
- Correcto funcionamiento de copias y asignaciones.
- Código modular, legible y bien estructurado.
- Sin fugas de memoria (memory leaks).

5. Estructura sugerida del proyecto

```
Biblioteca/
|-- main.cpp
|-- Libro.h / Libro.cpp
|-- Biblioteca.h / Biblioteca.cpp
+-- README.md / Enunciado.pdf
```