Facultad de Ingeniería
Universidad Nacional de Mar del Plata

Programación Orientada a Objetos
2024

Segundo Parcial

Teoría

1) Desarrolle una explicación académica de los principios fundamentales de la Programación Orientada a Objetos, ilustrando cada concepto con un ejemplo de implementación en C++ que demuestre su aplicación práctica: *Abstracción*, *Encapsulamiento*, *Herencia* y *Polimorfismo*.

2) Analice y fundamente detalladamente en qué circunstancias específicas es necesario declarar una función *friend* al sobrecargar operadores binarios en C++. Explique los escenarios donde esta decisión de diseño mejora la legibilidad y eficiencia del código.

3) Describa el mecanismo completo de manejo de *excepciones* en C++. Analice detalladamente qué sucede cuando se lanza una excepción, su propagación por la pila de llamadas, y explique cómo el patrón **RAII** garantiza la liberación de recursos ante una excepción inesperada.

4) Defina los conceptos de *alta cohesión* y *bajo acoplamiento* en diseño de software. Presente dos ejemplos de código C++ donde se violen estos principios, explique las consecuencias negativas y proponga refactorizaciones que mejoren el diseño.

5) Al *heredar de múltiples* clases, puede surgir un problema conocido como '**el diamante**'. Explicar en qué consiste este problema y sugerir dos estrategias para resolverlo.

6) Describa el mecanismo de *vtables* en C++ que posibilita el polimorfismo dinámico. Explique su implementación, sobrecarga de métodos virtuales, y las implicaciones de rendimiento y memoria.

7) ¿Qué es un *template* en C++ y para qué sirve? Explique qué problemas resuelve el uso de templates y cómo mejora la reutilización del código. Proporcione dos ejemplos prácticos de templates en C++: uno utilizando un template de función y otro utilizando un template de clase.

8) Explique qué es un *functor* (objeto función) en C++ y cómo se implementa. ¿Qué relación existe entre un functor y un predicado? Proporcione un ejemplo en C++ de la implementación de un functor y un ejemplo de cómo un predicado puede ser utilizado en un algoritmo estándar.

Práctica

Ejercicio 1

El programa proporcionado a continuación implementa un comportamiento genérico en la clase base Imprimible, pero tiene errores que impiden su correcta compilación y ejecución. Su tarea es resolver los problemas para que el código compile y funcione tal como está escrito en el main. Luego, deberá realizar una nueva implementación para que el main funcione sin cambios y produzca exactamente la misma salida que la versión original corregida.

```
#include <iostream>
#include <string>

template <typename T>
class Imprimible {
public:
    void imprimir() const {
        static_cast<const T*>(this)->imprimirImpl();
    }
protected:
    ~Imprimible() = default;
};

// Clase derivada para Persona
class Persona : public Imprimible<Persona> {
private:
    std::string nombre;
    int edad;

    void imprimirImpl() const {
```

```
        std::cout << "Persona - Nombre: " << nombre <<
", Edad: " << edad << std::endl;
    }

public:
    Persona(const std::string& nombre, int edad) :
nombre(nombre), edad(edad) {}
};

// Clase derivada para Producto
class Producto : public Imprimible<Producto> {
private:
    std::string nombre;
    double precio;

    void imprimirImpl() const {
        std::cout << "Producto - Nombre: " << nombre <<
", Precio: $" << precio << std::endl;
    }

public:
    Producto(const std::string& nombre, double precio) :
nombre(nombre), precio(precio) {}
};
```

```cpp
int main() {
    Persona p("Juan Perez", 30);
    Producto prod("Laptop", 999.99);

    p.imprimir();
    prod.imprimir();

    return 0;
}
```

Ejercicio 2

Diseñar e implementar un sistema de gestión de una biblioteca utilizando C++. El sistema deberá permitir:

- Crear y gestionar diferentes tipos de publicaciones: libros, revistas y periódicos, cada uno con sus propios atributos (título, autor, año de publicación, etc.).
- Almacenar las publicaciones en una estructura de datos adecuada.
- Persistir los datos de las publicaciones en archivos: tanto en formato texto como binario.
- Cargar los datos persistidos en la aplicación.
- Realizar operaciones básicas sobre las publicaciones: agregar, buscar, eliminar y modificar.

Se recomienda:

- Utilizar templates: Para crear estructuras de datos genéricas que puedan almacenar diferentes tipos de publicaciones.
- Manejar excepciones: Para detectar y gestionar errores de manera adecuada.
- Documentar el código: Utilizar comentarios para explicar el propósito de cada clase, método y variable.