



Ejercicio 26

Docente: Jimmy Nataniel Requena Llorentty

Materia: Programación III

Carrera: Ingeniería En Sistemas

Estudiantes: Joaquin Marcos Maita Flores

Santa Cruz – Bolivia

2025

¡ENSAMBLANDO NUESTRO AUTOMOVIL CON SU MOTOR! Parte II

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

// Clase 'Parte': Motor [EXPLICACIÓN: Clase base que representa el componente
Motor]
class Motor {
private:
    int caballosDeFuerza; // [NOTA: Almacena la potencia del motor en caballos
de fuerza]
    bool encendido;      // [NOTA: Estado actual del motor (true=encendido,
false=apagado)]

public:
    // Constructor con valor por defecto (150 HP) [OBSERVACIÓN: Inicializa ambos
atributos]
    Motor(int hp = 150) : caballosDeFuerza(hp), encendido(false) {
        std::cout << "  CONSTRUCTOR Motor: Creado motor de " << caballosDeFuerza
<< " HP." << std::endl;
        // [DETALLE: Mensaje muestra que se completó la construcción]
    }

    ~Motor() {
        std::cout << "  DESTRUCTOR Motor: Destruido motor de " <<
caballosDeFuerza << " HP." << std::endl;
        // [DETALLE: Se ejecuta automáticamente al destruir el objeto]
        std::cout << "\033[33mJoaquin Marcos Maita Flores.\033[0m" << std::endl;
    }

    // [MÉTODO: Controla el encendido del motor con validación de estado]
    void arrancar() {
        if (!encendido) {
            encendido = true;
            std::cout << "  Motor: ¡BRUM! Encendido." << std::endl;
        } else {
            std::cout << "  Motor: Ya estaba encendido." << std::endl;
        }
    }

    // [MÉTODO: Controla el apagado del motor con validación de estado]
    void detener() {
```

```
        if (encendido) {
            encendido = false;
            std::cout << "    Motor: ...silencio. Apagado." << std::endl;
        } else {
            std::cout << "    Motor: Ya estaba apagado." << std::endl;
        }
    }

    // [MÉTODO: Muestra el estado actual del motor]
    void mostrarEstado() const {
        std::cout << "    Estado del Motor: " << (encendido ? "Encendido" :
"Apagado")
                << ", HP: " << caballosDeFuerza << std::endl;
    }
};

// Clase 'Parte': Rueda [CONTEXTO: Segundo componente del automóvil]
class Rueda {
private:
    std::string tipo; // [CARACTERÍSTICA: Tipo de neumático]

public:
    // Constructor con valor por defecto ("Normal") [NOTA: Inicializa el tipo de
rueda]
    Rueda(std::string t = "Normal") : tipo(t) {
        std::cout << "    CONSTRUCTOR Rueda: Tipo = " << tipo << std::endl;
    }

    ~Rueda() {
        std::cout << "    DESTRUCTOR Rueda: Tipo = " << tipo << std::endl;
    }
};

// Clase 'Todo' o 'Contenedora': Automovil [RELACIÓN: Usa composición con Motor y
Rueda]
class Automovil {
private:
    std::string marca; // [ATRIBUTO: Identificador de marca]
    std::string modelo; // [ATRIBUTO: Identificador de modelo]
    Motor motorInterno; // [COMPONENTE: Motor del automóvil]
    Rueda ruedaDelantera; // [COMPONENTE: Rueda delantera]

public:
    // [INICIALIZACIÓN: Construye todos los miembros en orden]
    Automovil(std::string ma, std::string mo, int hpDelMotor)
```

```
        : marca(ma), modelo(mo), motorInterno(hpDelMotor),
ruedaDelantera("Deportiva") {
    std::cout << "CONSTRUCTOR Automovil: Ensamblado un " << marca << " " <<
modelo << std::endl;
}

~Automovil() {
    std::cout << "DESTRUCTOR Automovil: Desguazando el " << marca << " " <<
modelo << std::endl;

}

// [INTERFAZ: Expone funcionalidad del motor]
void encender() {
    std::cout << modelo << ": Intentando encender..." << std::endl;
    motorInterno.arrancar();
}

void apagar() {
    std::cout << modelo << ": Intentando apagar..." << std::endl;
    motorInterno.detener();
}

// [DIAGNÓSTICO: Muestra estado del sistema]
void verDiagnostico() const {
    std::cout << "Diagnóstico del " << modelo << ":" << std::endl;
    motorInterno.mostrarEstado();
}
};

// [DEMOSTRACIÓN: Función principal muestra el ciclo de vida completo]
int main() {
    std::cout << "--- Creando un Automovil en el Stack ---" << std::endl;
    Automovil miAuto("SuperMarca", "ModeloX", 200); // [ETAPA: Construcción]

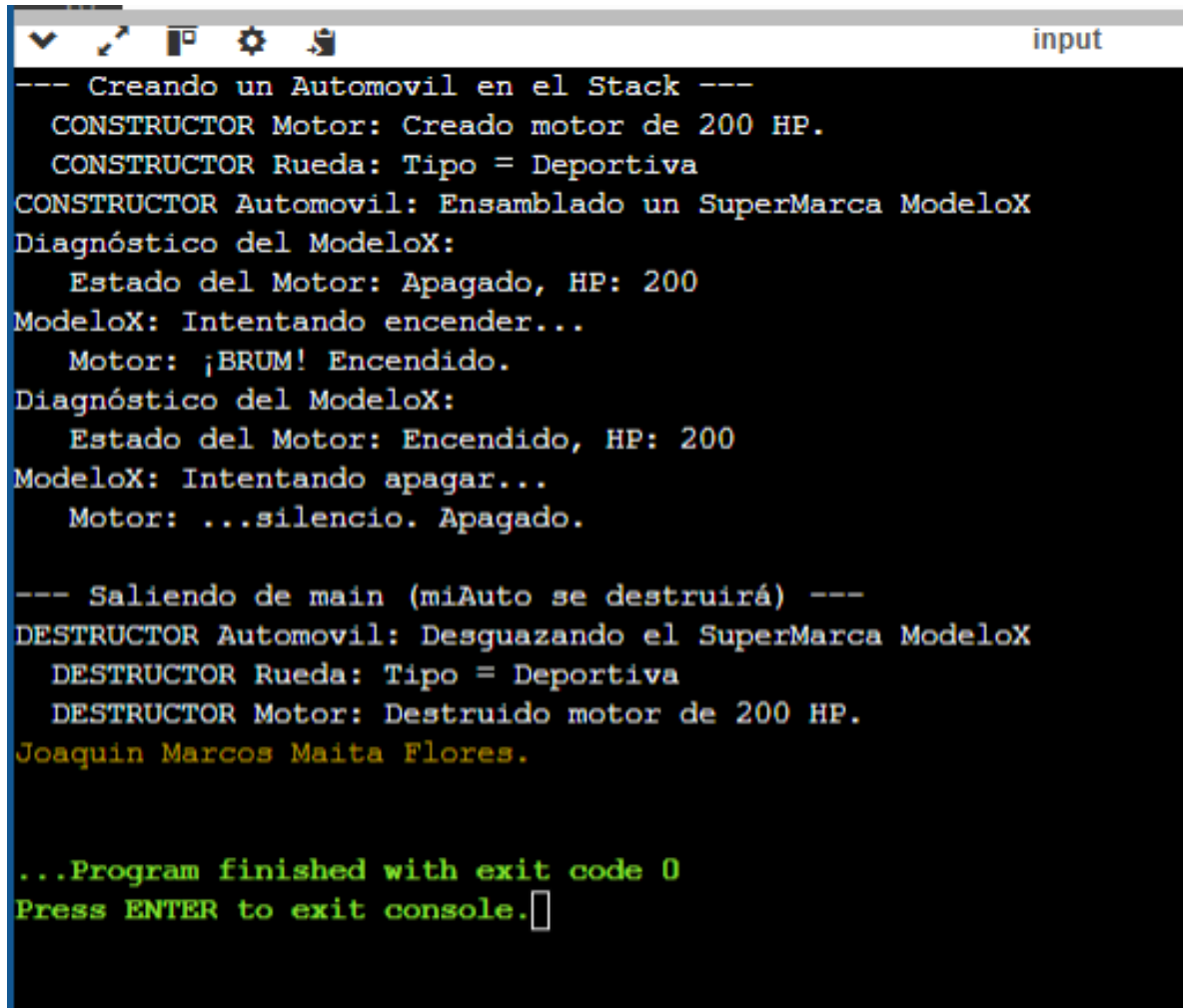
    miAuto.verDiagnostico(); // [USO: Consulta de estado]
    miAuto.encender();       // [USO: Operación básica]
    miAuto.verDiagnostico();
    miAuto.apagar();

    std::cout << "\n--- Saliendo de main (miAuto se destruirá) ---" << std::endl;
```

```
// [ETAPA: Destrucción automática al salir del ámbito]

return 0;
}
```

CÓDIGO EJECUTADO



```
--- Creando un Automovil en el Stack ---
CONSTRUCTOR Motor: Creado motor de 200 HP.
CONSTRUCTOR Rueda: Tipo = Deportiva
CONSTRUCTOR Automovil: Ensamblado un SuperMarca ModeloX
Diagnóstico del ModeloX:
    Estado del Motor: Apagado, HP: 200
ModeloX: Intentando encender...
    Motor: ¡BRUM! Encendido.
Diagnóstico del ModeloX:
    Estado del Motor: Encendido, HP: 200
ModeloX: Intentando apagar...
    Motor: ...silencio. Apagado.

--- Saliendo de main (miAuto se destruirá) ---
DESTRUCTOR Automovil: Desguazando el SuperMarca ModeloX
    DESTRUCTOR Rueda: Tipo = Deportiva
    DESTRUCTOR Motor: Destruído motor de 200 HP.
Joaquin Marcos Maita Flores.

...Program finished with exit code 0
Press ENTER to exit console.
```

Este ejemplo enseña:

1. Composición de Objetos:

- Automovil contiene instancias de Motor y Rueda (relación "tiene-un")
- Ejemplo práctico del principio de agregación en POO

2. Ciclo de Vida:

- Orden de construcción: miembros → contenedor
- Orden de destrucción: contenedor → miembros (demostrado con mensajes)

3. Encapsulamiento:

- Automovil actúa como interfaz, ocultando detalles de Motor/Rueda
- Métodos públicos (encender(), apagar()) delegan funcionalidad

4. Buenas Prácticas:

- Lista de inicialización en constructores
- Mensajes de depuración para seguimiento
- Valores por defecto en parámetros

Anexo

<https://onlinegdb.com/wWXcdm7-d>