Sistema de Gestión Polimórfica de Sensores IoT

Proyecto Final - Estructura de Datos

Ricardo Sebastián Balderas González
Universidad Politécnica de Victoria

31 de octubre de 2025

1. Introducción

Este proyecto implementa un sistema de gestión de sensores para una empresa de monitoreo de Infraestructura Crítica (IC). El objetivo principal es desarrollar un sistema flexible que permita manejar diferentes tipos de sensores y sus lecturas de manera unificada, superando las limitaciones de rigidez en el tipo de dato y estructura.

1.1. Objetivos

- Implementar un sistema polimórfico para gestión de sensores
- Manejar diferentes tipos de datos (int, float) de manera genérica
- Gestionar memoria dinámicamente sin usar STL
- Simular la captura de datos desde Arduino

2. Manual Técnico

2.1. Arquitectura del Sistema

El sistema se compone de tres capas principales:

- **Capa de Datos**: Implementación de listas enlazadas genéricas
- **Capa de Negocio**: Jerarquía de sensores y procesamiento
- **Capa de Presentación**: Interfaz de usuario en consola

2.2. Implementación Detallada

2.2.1. Lista Genérica

```
template <typename T>
class ListaSensor {
    struct Nodo {
        T dato;
        Nodo* siguiente;
        Nodo(const T& valor) : dato(valor), siguiente(
           nullptr) {}
    };

    Nodo* primero;

public:
    // Constructor y destructor
    ListaSensor() : primero(nullptr) {}
    ~ListaSensor();

    // Regla de los Tres
    ListaSensor(const ListaSensor& otra);
    ListaSensor& operator=(const ListaSensor& otra);

    // Operaciones b sicas
    void insertar(const T& valor);
    T calcularPromedio() const;
};
```

2.3. Gestión de Memoria

- **Destructor Virtual**: Implementado en SensorBase para garantizar la liberación correcta de memoria

- **Regla de los Tres**: Aplicada en ListaSensor para evitar fugas de memoria
- **Liberación en Cascada**: El sistema libera memoria jerárquicamente

2.4. Simulación Arduino

```
// Sketch de Arduino b sico
void loop() {
    // Simular lecturas
    Serial.println("T,45.3");
    delay(1000);
    Serial.println("P,80");
    delay(1000);
}
```

3. Diseño e Implementación

3.1. Jerarquía de Clases

El sistema se basa en una jerarquía polimórfica con las siguientes clases:

- **SensorBase (Clase Abstracta)**
  - Atributo protegido: char nombre[50]
  - Métodos virtuales puros:
    - procesarLectura()
    - imprimirInfo()
    - esTemperatura()
- **Clases Derivadas**
  - SensorTemperatura: Maneja lecturas float
  - SensorPresion: Maneja lecturas int

3.2. Estructura de Datos

3.2.1. Lista Genérica (ListaSensor¡T¿)

```
template <typename T>
class ListaSensor {
    struct Nodo {
        T dato;
        Nodo* siguiente;
    };
    // ...
};
```

3.2.2. Gestión de Memoria

Implementación de la Regla de los Tres:

- Destructor virtual en SensorBase
- Constructor de copia en ListaSensor
- Operador de asignación en ListaSensor

4. Implementación

4.1. Principales Componentes

4.1.1. GestorSensores

Esta clase maneja la lista principal de sensores y coordina todas las operaciones:

- Agregar sensores
- Registrar lecturas
- Procesar datos
- Liberar memoria

4.1.2. Procesamiento de Datos

Cada tipo de sensor implementa su propia lógica de procesamiento:

- SensorTemperatura: Calcula promedios y elimina valores mínimos
- SensorPresion: Calcula promedios de lecturas

5. Pruebas y Resultados

5.1. Casos de Prueba

Se realizaron pruebas exhaustivas de:

- Creación y eliminación de sensores
- Registro de lecturas
- Procesamiento polimórfico
- Gestión de memoria

5.2. Resultados y Evidencias

5.2.1. Pruebas Realizadas

- **Creación de Sensores**: Se verificó la creación correcta de sensores T-001 y P-105
- **Registro de Datos**: Se comprobó el registro de lecturas float (45.3, 42.1) y enteras (80, 85)
- **Procesamiento**: Se validó el cálculo de promedios y eliminación de valores mínimos
- **Liberación de Memoria**: Se confirmó la liberación correcta de todos los recursos

5.2.2. Demostración de Requisitos Cumplidos

- **Polimorfismo**: Implementado a través de la clase base SensorBase y sus derivadas
- **Templates**: Lista genérica ListaSensor¡T¿maneja diferentes tipos de datos
- **Gestión de Memoria**: Implementación completa de la Regla de los Tres
- **Arduino**: Simulación funcional de lecturas mediante puerto serial
- **CMake**: Sistema completamente redistribuible

6. Conclusiones y Aprendizajes

Durante el desarrollo de este proyecto, logré:

- Comprender la importancia del polimorfismo en el diseño de sistemas extensibles
- Implementar gestión manual de memoria sin depender de la STL
- Crear templates funcionales para manejar diferentes tipos de datos
- Integrar hardware (Arduino) con software (C++)
- Documentar código usando Doxygen

7. Referencias

- Stroustrup, B. (2013). The C++ Programming Language, 4th Edition
- Documentación oficial de CMake
- Arduino Programming Reference