Sistema Decodificador de Protocolo Industrial (PRT-7)

Implementación Polimórfica con Listas Enlazadas

Angel Gabriel Coronado Sánchez
`Matrícula: 2430124`
Estructuras de Datos

6 de Noviembre del 2025

Índice

1. Introducción

1.1. Contexto del Problema

Una firma de ciberseguridad industrial necesita descifrar un protocolo de transmisión interceptado desde sensores industriales simulados mediante ESP32. El protocolo identificado como **PRT-7** no transmite mensajes encriptados directamente, sino *instrucciones para construir el mensaje* dinámicamente.

El protocolo opera con dos tipos de tramas:

1. **Tramas de Carga (LOAD):** Contienen fragmentos de datos (caracteres individuales) que deben almacenarse secuencialmente.
2. **Tramas de Mapeo (MAP):** Contienen instrucciones de rotación que modifican un disco de cifrado circular.

La complejidad radica en que las tramas MAP modifican el estado del cifrado mientras se procesan las tramas LOAD, haciendo que el significado de cada carácter cambie dinámicamente.

1.2. Objetivo del Proyecto

Desarrollar un **Sistema Decodificador Polimórfico en C++** que gestione:

- Jerarquía de clases polimórfica para diferentes tipos de tramas
- Lista doblemente enlazada para almacenar mensaje decodificado
- Lista circular doblemente enlazada como rotor de cifrado
- Comunicación serial con ESP32
- Gestión manual de memoria sin STL

1.3. Alcance

El sistema incluye:

- Recepción de datos desde ESP32 por puerto serial
- Clase base abstracta `TramaBase` con clases derivadas
- Implementación manual de listas enlazadas
- Parser de protocolo PRT-7
- Sistema redistribuible con CMake
- Documentación con Doxygen

2. Manual Técnico

2.1. Diseño

2.1.1. Clase Base Abstracta: TramaBase

Propósito: Define la interfaz común para todas las tramas del protocolo.
Características:

- Método virtual puro: `procesar()`
- Destructor virtual para polimorfismo correcto

```
class TramaBase {
public:
    virtual void procesar(ListaDeCarga* carga,
                          RotorDeMapeo* rotor) = 0;
    virtual ~TramaBase() {}
};
```

Listing 1: Definición de TramaBase

2.1.2. Clases Derivadas Concretas

TramaLoad:

- Almacena un carácter
- Procesamiento: consulta rotor y guarda carácter decodificado

```
class TramaLoad : public TramaBase {
private:
    char dato;
public:
    TramaLoad(char d);
    void procesar(ListaDeCarga* carga,
                  RotorDeMapeo* rotor) override;
};
```

Listing 2: Clase TramaLoad

TramaMap:

- Almacena valor de rotación
- Procesamiento: rota el rotor N posiciones

```
class TramaMap : public TramaBase {
private:
    int valorRotacion;
public:
    TramaMap(int valor);
```

```
    void procesar(ListaDeCarga* carga,
                  RotorDeMapeo* rotor) override;
};
```

Listing 3: Clase TramaMap

2.1.3. Lista Doblemente Enlazada: ListaDeCarga

Propósito: Almacenar el mensaje decodificado en orden.
Estructura de Nodo:

```
class NodoDoble {
public:
    char dato;
    NodoDoble* anterior;
    NodoDoble* siguiente;
    NodoDoble(char d);
};
```

Listing 4: NodoDoble

Operaciones:

- `insertarAlFinal(char dato)`
- `imprimirMensaje()`
- `obtenerMensaje(char* buffer, int tamano)`

2.1.4. Lista Circular: RotorDeMapeo

Propósito: Disco de cifrado circular que mapea caracteres.
Estructura de Nodo:

```
class NodoCircular {
public:
    char dato;
    NodoCircular* anterior;
    NodoCircular* siguiente;
    NodoCircular(char d);
};
```

Listing 5: NodoCircular

Operaciones:

- `rotar(int n)`: Rota el rotor N posiciones
- `getMapeo(char entrada)`: Obtiene carácter mapeado
- `getCaracterCabeza()`: Para debugging

2.2. Desarrollo

2.2.1. Tecnologías Utilizadas

Componente	Tecnología	Propósito
Lenguaje	C++ (C++14)	Lenguaje principal
Compilador	MinGW/GCC	Compilación
Build System	CMake	Gestión multiplataforma
Documentación	Doxygen	Generación de docs
Hardware	ESP32	Transmisión serial
Comunicación	WinAPI Serial	Puerto COM

Cuadro 1: Stack tecnológico

2.2.2. Decisiones de Diseño Clave

1. Polimorfismo mediante Clase Base Abstracta
Justificación:

- Procesar tramas heterogéneas uniformemente
- Sistema extensible a nuevos tipos de trama
- Binding dinámico ejecuta lógica específica

2. Listas Enlazadas Manuales
Justificación:

- Requisito del proyecto (sin STL)
- Control total de memoria
- Demostración de dominio de punteros

3. Lista Circular para Rotor
Justificación:

- Rotación eficiente sin mover datos
- Navegación bidireccional
- Modelo natural de disco de cifrado

4. Comunicación Serial con WinAPI
Justificación:

- Control directo del hardware
- Sin dependencias externas
- Configuración precisa de parámetros

2.2.3. Arquitectura del Código

```
proyecto/
 main.cpp
 Decodificador.h/.cpp
 TramaBase.h
 TramaLoad.h/.cpp
 TramaMap.h/.cpp
 ListaDeCarga.h/.cpp
 RotorDeMapeo.h/.cpp
 NodoDoble.h/.cpp
 NodoCircular.h/.cpp
 Serial.h/.cpp
 CMakeLists.txt
 Doxyfile
```

Flujo de ejecución:

1. Inicializar `ListaDeCarga` y `RotorDeMapeo`
2. Abrir puerto serial con ESP32
3. Leer líneas (`L,X` o `M,N`)
4. Parser crea trama correspondiente
5. Ejecutar `procesar()` polimórficamente
6. Liberar memoria de trama
7. Mostrar mensaje decodificado

2.3. Componentes

2.3.1. TramaBase

Define contrato común. Destructor virtual crítico para evitar fugas de memoria polimórficas.

2.3.2. TramaLoad

Lógica de procesamiento:

```
void TramaLoad::procesar(ListaDeCarga* carga,
                         RotorDeMapeo* rotor) {
    char decodificado = rotor->getMapeo(dato);
    carga->insertarAlFinal(decodificado);
}
```

2.3.3. TramaMap

Lógica de procesamiento:

```
void TramaMap::procesar(ListaDeCarga* carga,
                        RotorDeMapeo* rotor) {
    rotor->rotar(valorRotacion);
}
```

2.3.4. RotorDeMapeo

Método de rotación:

```
void RotorDeMapeo::rotar(int n) {
    if (n > 0) {
        for (int i = 0; i < n; i++)
            cabeza = cabeza->siguiente;
    } else if (n < 0) {
        for (int i = 0; i > n; i--)
            cabeza = cabeza->anterior;
    }
}
```

Método de mapeo:

```
char RotorDeMapeo::getMapeo(char entrada) {
    int pos = entrada - 'A';
    NodoCircular* nodo = cabeza;
    for (int i = 0; i < pos; i++)
        nodo = nodo->siguiente;
    return nodo->dato;
}
```

2.3.5. Decodificador

Parser de protocolo:

```
TramaBase* Decodificador::parsearLinea(const char* linea) {
    char tipo;
    char buffer[256];

    if (sscanf(linea, "%c,%s", &tipo, buffer) != 2)
        return nullptr;

    if (tipo == 'L')
        return new TramaLoad(buffer[0]);
    else if (tipo == 'M')
        return new TramaMap(atoi(buffer));

    return nullptr;
}
```

2.3.6. Serial

Encapsula comunicación WinAPI para lectura desde puerto COM.

3. Pantallazos de Implementación

3.1. Inicio de la ejecución

```
PS C:\Users\as_co\C\ds-unidad-2-actividad-2-listasenlazadas-AngelCS0124> build\DecodificadorPRT7.exe
=== DECODIFICADOR PRT-7 ===
Iniciando Decodificador PRT-7...
Conectando a puerto COM6...
Conectando a COM6...
Conexion establecida con COM6 a 9600 baudios
Conexion establecida con COM6
Esperando datos del ESP32...
```

Figura 1: Comprobación de la conexión ESP32

3.2. Procesamiento de tramas

```
Trama 2: 'L,I'
Trama recibida: [L,I] -> Procesando... -> Fragmento 'I' decodificado como 'I'. Mensaje: [I]
Trama 3: 'M,-2'
Trama recibida: [M,-2] -> Procesando... -> ROTANDO ROTOR -2. (Antes 'A' se mapeaba a 'A', ahora a 'Y')
Trama 4: 'L,V'
Trama recibida: [L,V] -> Procesando... -> Fragmento 'V' decodificado como 'T'. Mensaje: [I][T]
Trama 5: 'L,K'
Trama recibida: [L,K] -> Procesando... -> Fragmento 'K' decodificado como 'I'. Mensaje: [I][T][I]
Trama 6: 'M,4'
Trama recibida: [M,4] -> Procesando... -> ROTANDO ROTOR +4. (Antes 'Y' se mapeaba a 'Y', ahora a 'C')
Trama 7: 'L,G'
Trama recibida: [L,G] -> Procesando... -> Fragmento 'G' decodificado como 'I'. Mensaje: [I][T][I][I]
Trama 8: 'L,B'
Trama recibida: [L,B] -> Procesando... -> Fragmento 'B' decodificado como 'D'. Mensaje: [I][T][I][I][D]
Trama 9: 'FIN'
Fin de transmision
```

Figura 2: Decodificación de tramas en tiempo real

3.3. Mensaje final

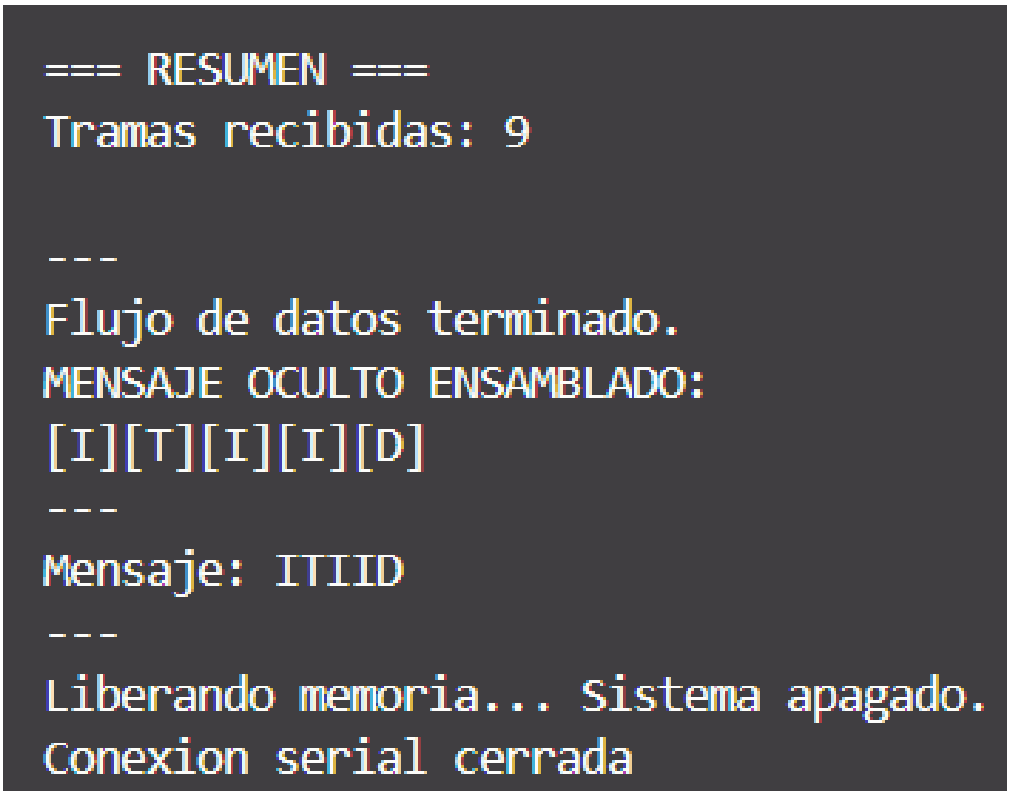

Figura 3: Mensaje oculto decodificado

4. Conclusiones

4.1. Objetivos Alcanzados

Jerarquía Polimórfica: Implementación exitosa de `TramaBase` con clases derivadas `TramaLoad` y `TramaMap`, demostrando uso correcto de métodos virtuales puros y destructores virtuales.

Listas Enlazadas Manuales: Desarrollo desde cero de lista doblemente enlazada (`ListaDeCarga`) y lista circular (`RotorDeMapeo`), cumpliendo requisito de no usar STL.

Protocolo PRT-7: Decodificación correcta de mensajes procesando tramas de carga y mapeo secuencialmente con rotor de mapeo dinámico.

Comunicación Serial: Conexión exitosa con ESP32 mediante WinAPI, con parseo correcto del formato de protocolo.

Gestión de Memoria: Implementación de destructores apropiados sin fugas de memoria, garantizando limpieza polimórfica correcta.

Sistema Redistribuible: CMake permite compilación multiplataforma con configuración unificada.

Documentación: Generación de documentación técnica completa mediante Doxygen con estándares profesionales.

— Fin del Reporte —