NOMBRE DEL ALUMNO:

MYRANDA BELEN INFANTE CASTILLO

NOMBRE DE LA MATERIA:

ESTRUCTURA DE DATOS

NOMBRE DE LA ACTIVIDAD:

Actividad 2 - Decodificador de Protocolo Industrial (PRT-7)

Nombre del profesor:

DR. SAID POLANCO MARTAGÓN (Dr. en Artes Oscuras)

INTRODUCCIÓN

Este proyecto implementa un Decodificador de Protocolo Industrial (PRT-7) que demuestra el uso avanzado de conceptos de Programación Orientada a Objetos en C++ **SIN USAR LA STL**.

Problema a Resolver: Una firma de ciberseguridad industrial necesita decodificar un protocolo de ensamblaje de mensajes ocultos llamado PRT-7. El protocolo transmite dos tipos de tramas desde un dispositivo Arduino: tramas de carga (LOAD) que contienen fragmentos de datos, y tramas de mapeo (MAP) que contienen instrucciones para reordenar dinámicamente estos fragmentos.

Objetivos:

- Crear jerarquía de clases polimórfica con herencia
- Implementar listas doblemente enlazadas manualmente
- Implementar listas circulares doblemente enlazadas manualmente
- Gestionar memoria con la Regla de los Tres
- Capturar datos en tiempo real desde Arduino
- **NO usar ninguna librería STL**

MANUAL TÉCNICO

DISEÑO DEL SISTEMA:

El sistema utiliza los siguientes patrones:

Patrón	Aplicación
Herencia	TramaBase → TramaLoad, TramaMap
Polimorfismo	Método virtual procesar()

Listas Manuales	ListaDeCarga (doblemente enlazada) RotorDeMapeo (circular doblemente enlazada)
RAII	Gestión automática en destructores
SIN STL	Solo printf, scanf, funciones de C puro

COMPONENTES PRINCIPALES

1. TramaBase (Clase Abstracta)

- Métodos virtuales puros: procesar()
- Destructor virtual para polimorfismo correcto
- Define la interfaz común para todas las tramas

2. ListaDeCarga (Lista Doblemente Enlazada)

- Estructura NodoCarga para almacenar datos tipo char
- Operaciones: insertarAlFinal(), imprimirMensaje()
- Implementa Regla de los Tres
- **Sin STL:** usa printf() y punteros manuales

3. TramaLoad (Clase Derivada)

1. Maneja tramas de carga tipo L,X
2. Almacena un carácter y lo decodifica usando el rotor
3. Procesamiento: obtiene el mapeo del rotor y agrega a la lista de carga
4. **Sin STL:** usa printf() en lugar de cout

4. RotorDeMapeo (Lista Circular Doblemente Enlazada)

1. Estructura NodoRotor para almacenar el alfabeto A-Z
2. Operaciones: rotar(), getMapeo()
3. Gestión de punteros previo y siguiente para navegación circular
4. **Sin STL:** usa strcmp() y punteros manuales

ARQUITECTURA

Capas del Sistema:

1. Comunicación - SerialPort lee desde Arduino

2. Lógica - Clases de tramas y procesamiento polimórfico

3. Datos - Listas enlazadas manuales (ListaDeCarga y RotorDeMapeo)

4. Decodificación - Proceso de mapeo dinámico con el rotor

Sin STL:

• No usa iostream, string, vector, list

• Usa cstdio (printf, scanf)

• Usa cstring (strcmp, strncpy, strtok)

• Usa cstdlib (atoi)

• Usa NULL en lugar de nullptr

FLUJO DEL PROGRAMA

1. Inicialización

- Se crea la ListaDeCarga vacía
- Se crea el RotorDeMapeo con alfabeto A-Z
- Se establece conexión con puerto serial

2. Bucle Principal

- Leer línea del puerto serial
- Parsear la línea (determinar si es L o M)
- Instanciar objeto TramaLoad o TramaMap
- Ejecutar trama->procesar(carga, rotor)
- Liberar memoria de la trama
- Repetir hasta fin de transmisión

3. Decodificación

- TramaLoad: obtiene mapeo del rotor e inserta en lista
- TramaMap: rota el rotor N posiciones
- El estado del rotor afecta cómo se decodifican las tramas subsecuentes

4. Finalización

- Imprimir mensaje completo decodificado
- Liberar memoria de todas las estructuras
- Cerrar puerto serial

GESTIÓN DE MEMORIA

El proyecto implementa correctamente la gestión manual de memoria:

1. Constructor de Copia

- ListaDeCarga: copia profunda de todos los nodos
- RotorDeMapeo: copia profunda manteniendo estructura circular

2. Operador de Asignación

- Verifica auto-asignación
- Libera memoria existente
- Realiza copia profunda

3. Destructor

- ListaDeCarga: recorre y libera cada nodo

• RotorDeMapeo: maneja correctamente la circularidad

• TramaBase: destructor virtual para limpieza polimórfica

LÓGICA DEL ROTOR DE MAPEO

El RotorDeMapeo es una lista circular doblemente enlazada que contiene el alfabeto:

Estructura: A ↔ B ↔ C ↔ ... ↔ Z ↔ (vuelve a A)

• **Puntero cabeza:** marca la posición de referencia actual

• **rotar(N):** mueve cabeza N posiciones (+ o - en la lista circular)

• **getMapeo(char in):**

1. Encuentra posición de 'in' en la lista
2. Calcula desplazamiento relativo a cabeza
3. Aplica desplazamiento para obtener carácter mapeado
4. Retorna el carácter decodificado

EJEMPLO DE DECODIFICACIÓN

Estado inicial: cabeza → A

Trama L,H → procesar()

• rotor->getMapeo('H') con cabeza en A

• H está a 7 posiciones de A

• Mapeo directo: H → H

• Lista: [H]

Trama M,2 → procesar()

• rotor->rotar(2)

• cabeza se mueve: A → B → C

• Ahora cabeza → C

Trama L,A → procesar()

• rotor->getMapeo('A') con cabeza en C

- A está a -2 posiciones de C (o +24 circular)
- Con el desplazamiento: A → C
- Lista: [H][C]

CONCLUSIONES

Para este proyecto se utilize: :

1. POO avanzada: herencia, polimorfismo, clases abstractas

2. Estructuras de datos complejas: listas doblemente enlazadas y circulares

3. Gestión manual de memoria: sin fugas, destructores virtuales

4. Comunicación en tiempo real: lectura de puerto serial

5. Parsing de protocolos: interpretación de cadenas de texto

El sistema logra decodificar exitosamente mensajes ocultos transmitidos mediante el protocolo PRT-7, demostrando que el manejo correcto de estructuras dinámicas y polimorfismo permite resolver problemas complejos de ciberseguridad industrial.

El desafío principal fue mantener sincronizadas las operaciones de rotación del mapeo con la decodificación de fragmentos, lo cual requirió un entendimiento profundo de cómo el estado de una estructura afecta el procesamiento de otra.