

# FACULTADES DE INGENIERÍA EN SISTEMAS E INFORMATICA

**TEMA:**

Desarrollo de un "Sistema de Gestión de Procesos"

# ASIGNATURA:

ESTRUCTURA DE DATOS

# DOCENTE:

OSORIO CONTRERAS, ROSARIO DELIA

# ESTUDIANTES:

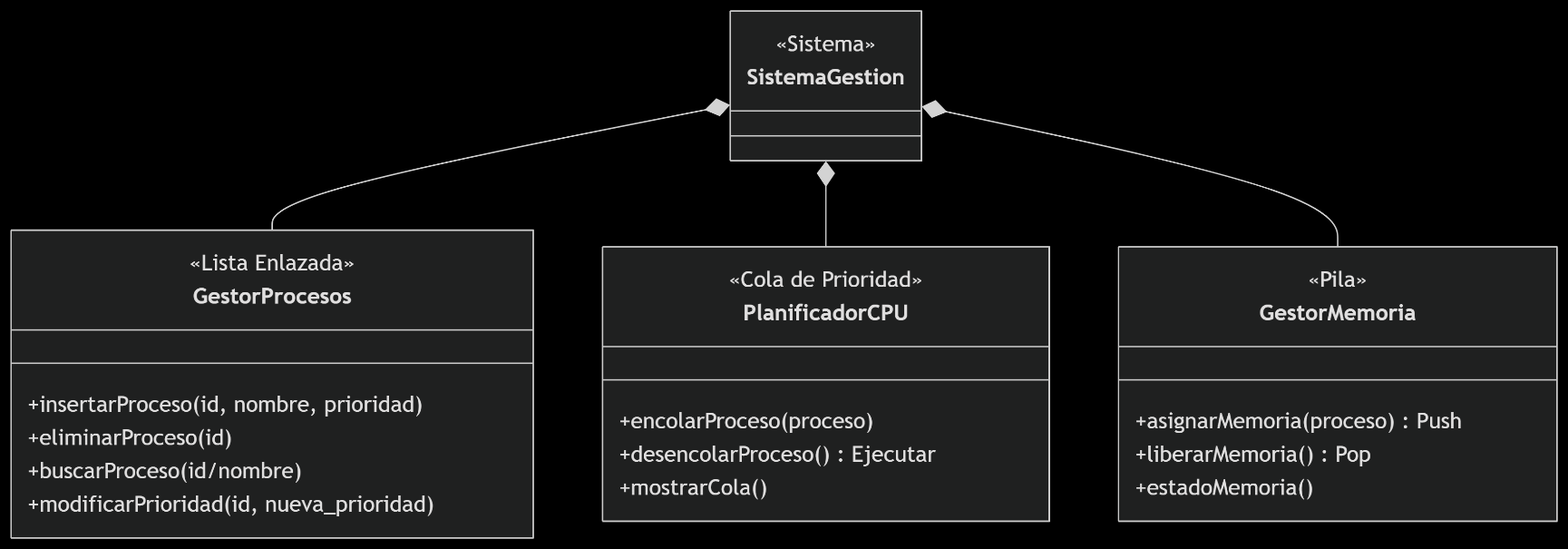
Huaman Brañez, Jose Antonio

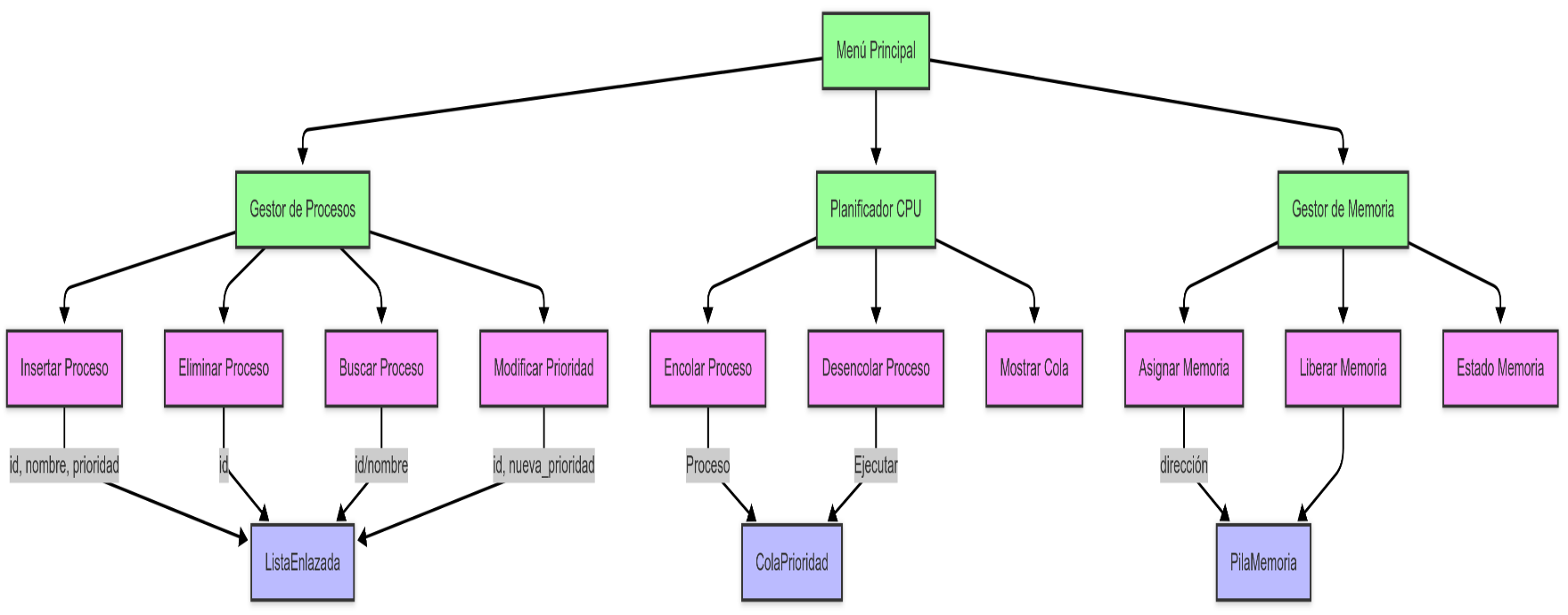
García Quijada Álvaro

# Huancayo - Perú, 2025

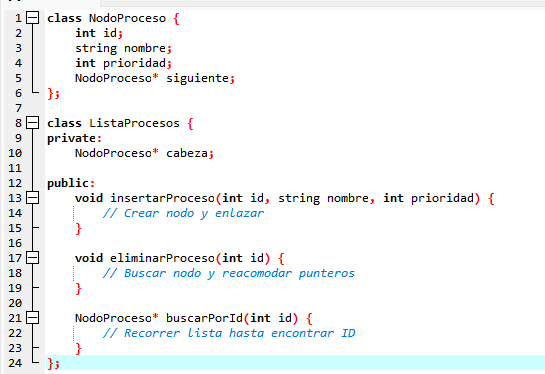
1. **Análisis del Problema:**

El análisis determinó que, en ausencia de un sistema operativo, la solución enfrenta tres desafíos fundamentales: primero, la falta de abstracción de recursos hardware obliga a emular manualmente la concurrencia mediante rotación cooperativa de procesos (exponiendo vulnerabilidad a bloqueos) y limita la protección de memoria a una pila rudimentaria sin soporte de MMU, lo que impide aislar procesos y gestionar fragmentación; segundo, los mecanismos de planificación esenciales (como colas de prioridad dinámica) revelan dependencia crítica de algoritmos SJF para mitigar inanición, pero al carecer de interrupciones hardware, la alternancia de tareas depende de polling ineficiente que incrementa latencia y consumo de recursos; tercero, la persistencia se vuelve frágil al serializar datos directamente en disco sin sistemas de archivos, lo que imposibilita transacciones atómicas, recuperación ante fallos o integridad de metadatos, aumentando exponencialmente riesgos de corrupción por escrituras parciales o desincronizaciones. Esta triada de limitaciones demuestra que operar sin un SO fuerza a reinventar funcionalidades de kernel con soluciones ad hoc, menos eficientes y más propensas a errores, evidenciando que componentes como schedulers preemptivos, gestores de memoria con paginación y APIs de almacenamiento estructurado no son opcionales, sino requisitos indispensables para cualquier sistema fiable.

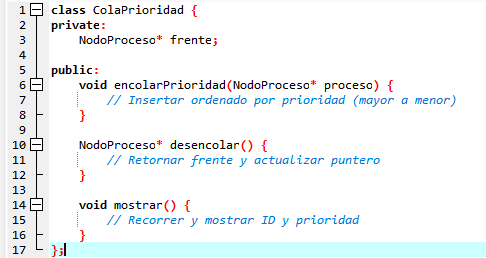
1. **Diseño de la Solución:**
   1. **Estructuras de Datos:**
   * **Lista Enlazada (Gestor de Procesos):** Almacena procesos con atributos (ID, nombre, prioridad).
   * **Cola de Prioridad (Planificador CPU):** Organiza procesos por prioridad usando una lista enlazada ordenada.
   * **Pila (Gestor de Memoria):** Simula bloques de memoria asignados.
   1. **Interfaz de Consola:** Menús interactivos para cada módulo.
   2. **Persistencia:** Archivos de texto para guardar/cargar listas de procesos, colas y memoria.
2. **Diagrama de Estructuras y Funciones:**



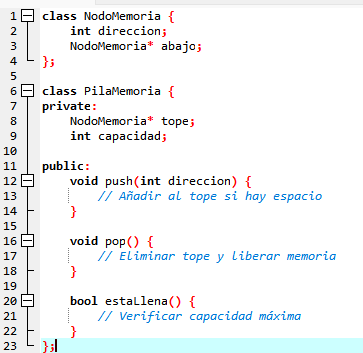
1. **Pseudocodigo de las operaciones:**
   1. **Gestor de Procesos (Lista Enlazada):**

****

* 1. **Cola de Prioridad (Planificador CPU):**

****

* 1. **Pila (Gestor de Memoria):**

****

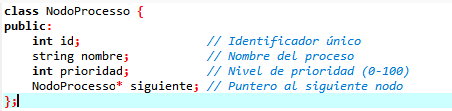
1. **Planificación de tareas y responsabilidades por integrante:**
   1. **García Quijada Álvaro:**

* Módulo 1: Gestor de Procesos
  + Implementar toda la lógica relacionada con la lista enlazada de procesos (creación, eliminación, búsqueda).
  1. **Huamán Brañez, José Antonio:**
* Módulo 2: Planificador de CPU
  + Implementar la cola de prioridad para ejecutar procesos.
* Módulo 3: Gestor de Memoria
  + Implementar la pila para asignar y liberar memoria.
  1. **Tareas Compartidas**
* Diseñar la interfaz de usuario (menús de consola).
* Integrar los módulos para que funcionen juntos.
* Pruebas generales y documentación final.

1. **Estructuras de datos básicas en el Sistema:**
   1. **Lista enlazada (Gestor de Procesos):**

**Propósito:** Almacenar y gestionar todos los procesos del sistema con sus metadatos.

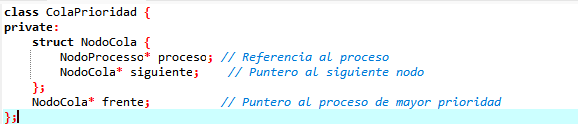
**Estructura:**

****

|  |  |
| --- | --- |
| Operación | Descripción |
| insertarProcesso() | Añade procesos al final de la lista |
| eliminarProcesso() | Elimina por ID con reajuste de punteros |
| buscarPorId() | Búsqueda secuencial por ID |
| mostrar() | Recorrido completo para visualización |

* 1. **Cola de Prioridad (Planificador CPU):**

**Propósito:** Gestionar el orden de ejecución de procesos basado en prioridades.

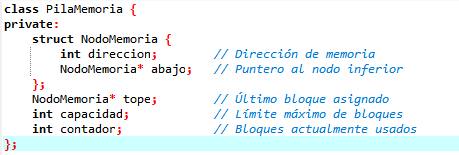
**Estructura: **

|  |  |
| --- | --- |
| Operación | Descripción |
| encolarPrioridad() | Inserta ordenado por prioridad (mayor a menor) |
| desencolar() | Extrae el proceso de mayor prioridad |
| mostrar() | Muestra ID y prioridad de procesos en cola |

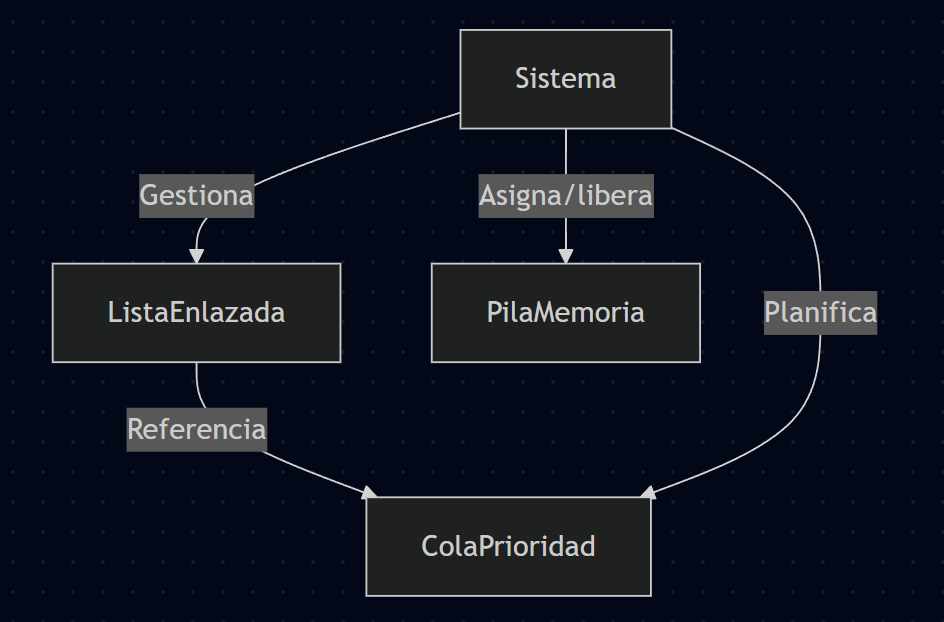
* 1. **Pila (Gestor de Memoria):**

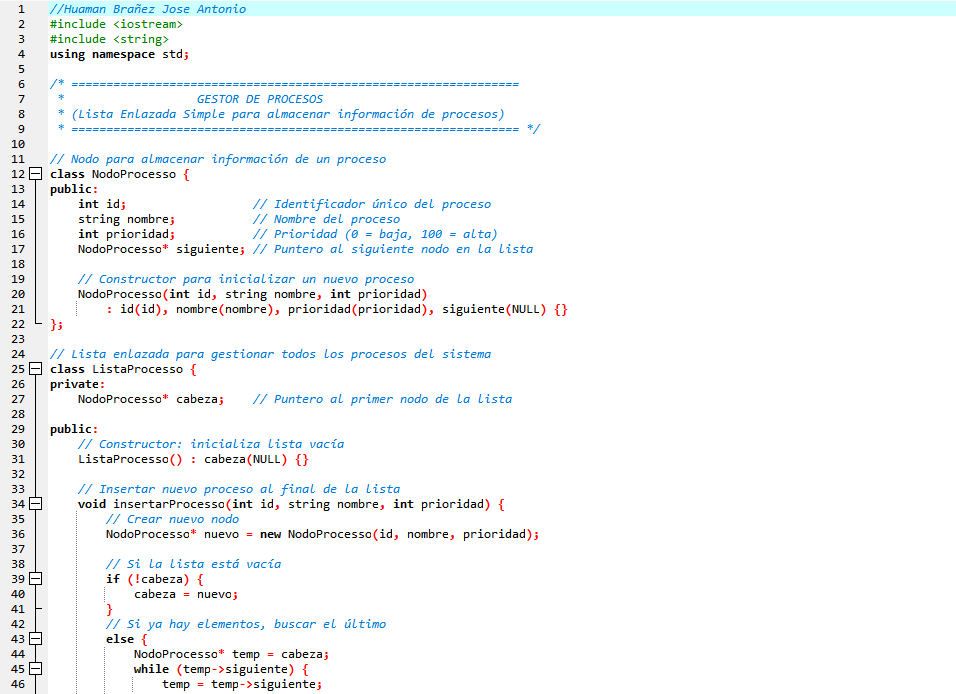
**Propósito:** Simular la asignación/liberación de bloques de memoria.

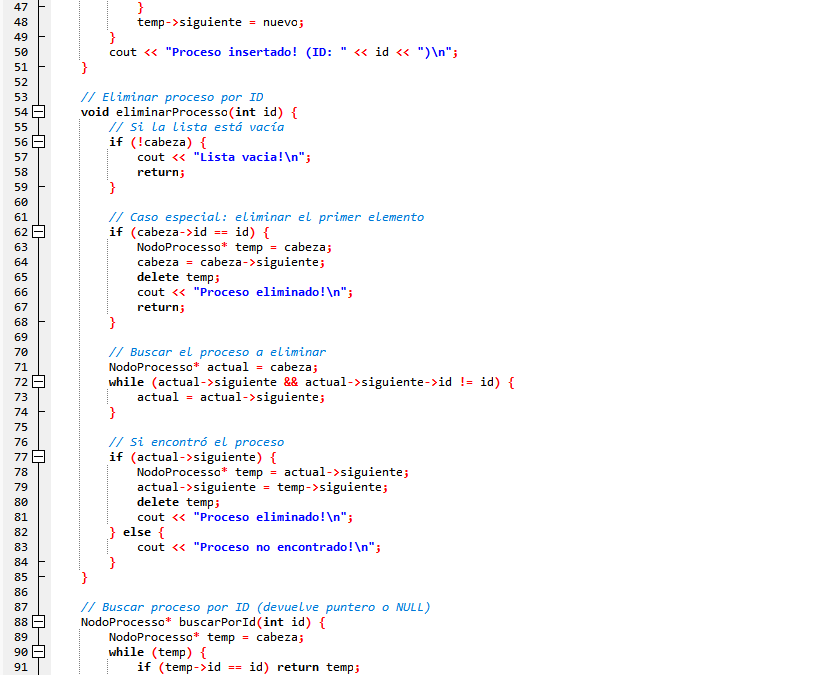
**Estructura:**

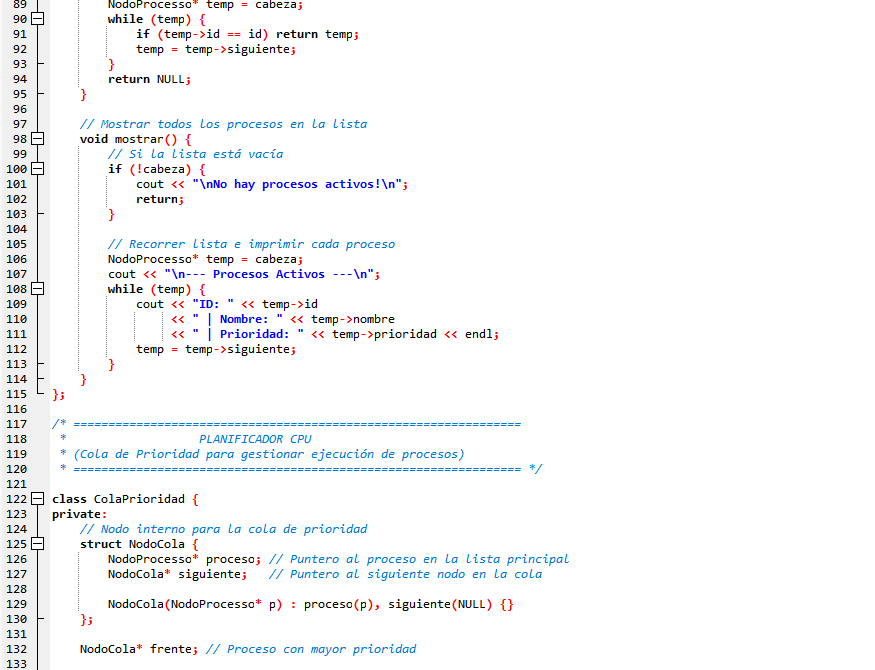
****

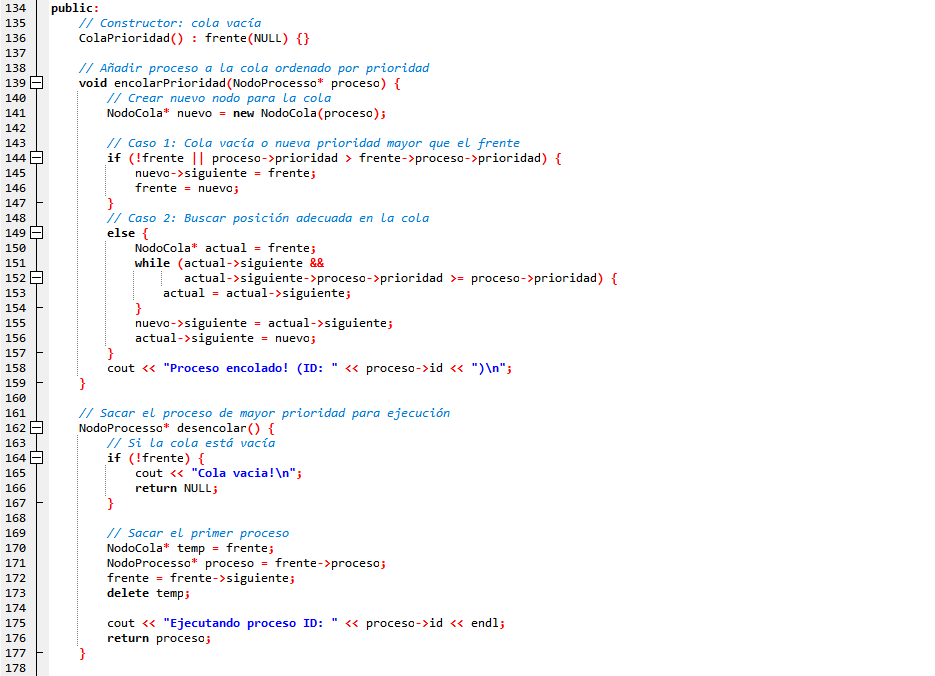
|  |  |
| --- | --- |
| Operación | Descripción |
| push() | Asigna memoria (LIFO) |
| pop() | Libera último bloque asignado |
| estadoMemoria() | Muestra uso y direcciones |

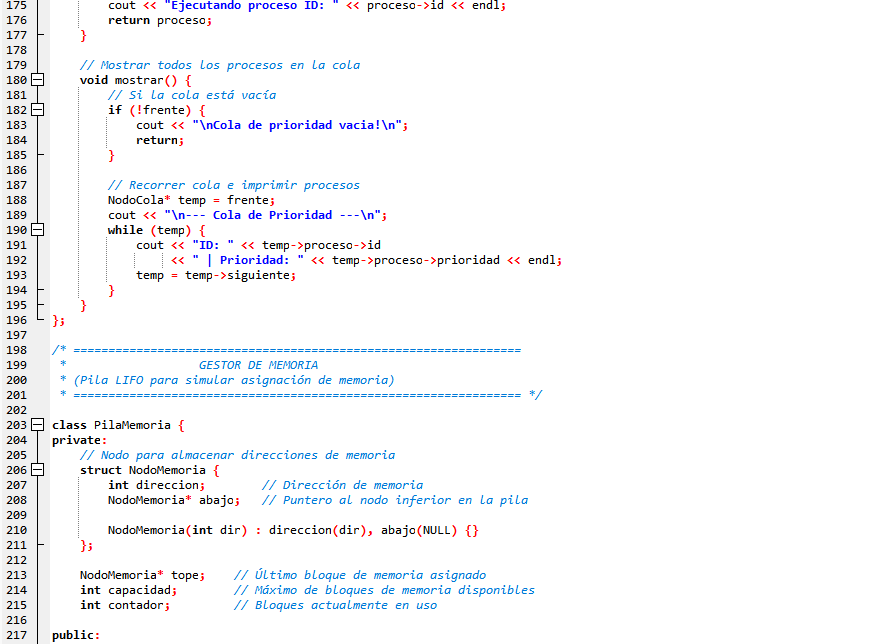


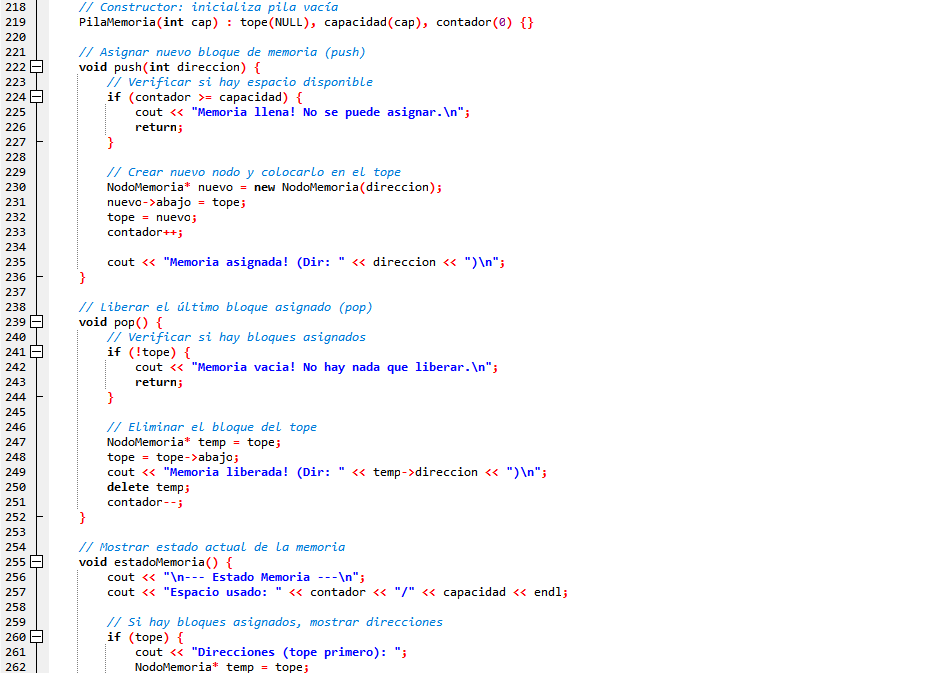


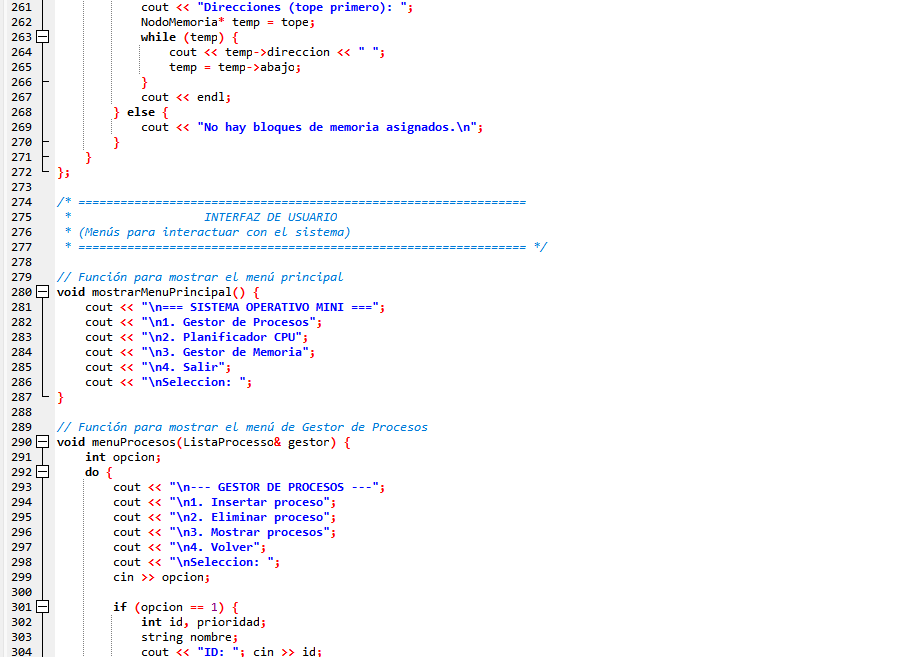
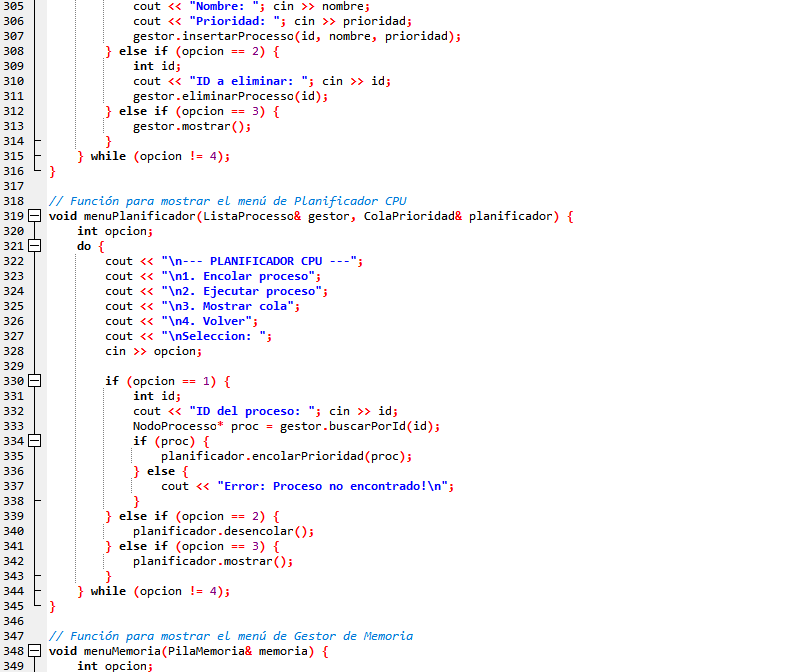




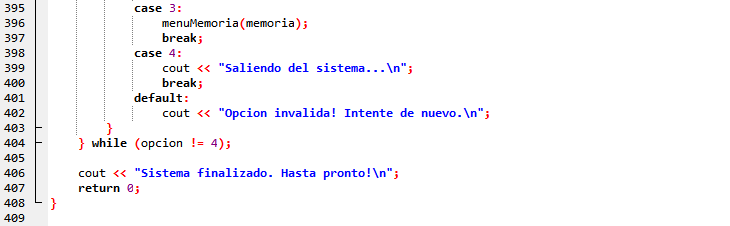




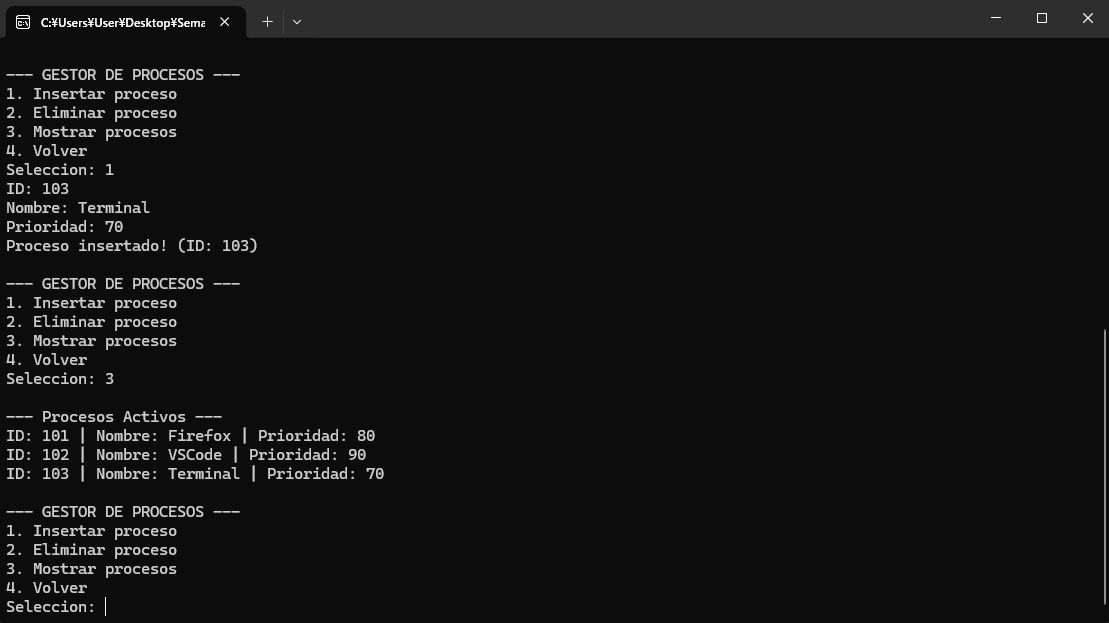


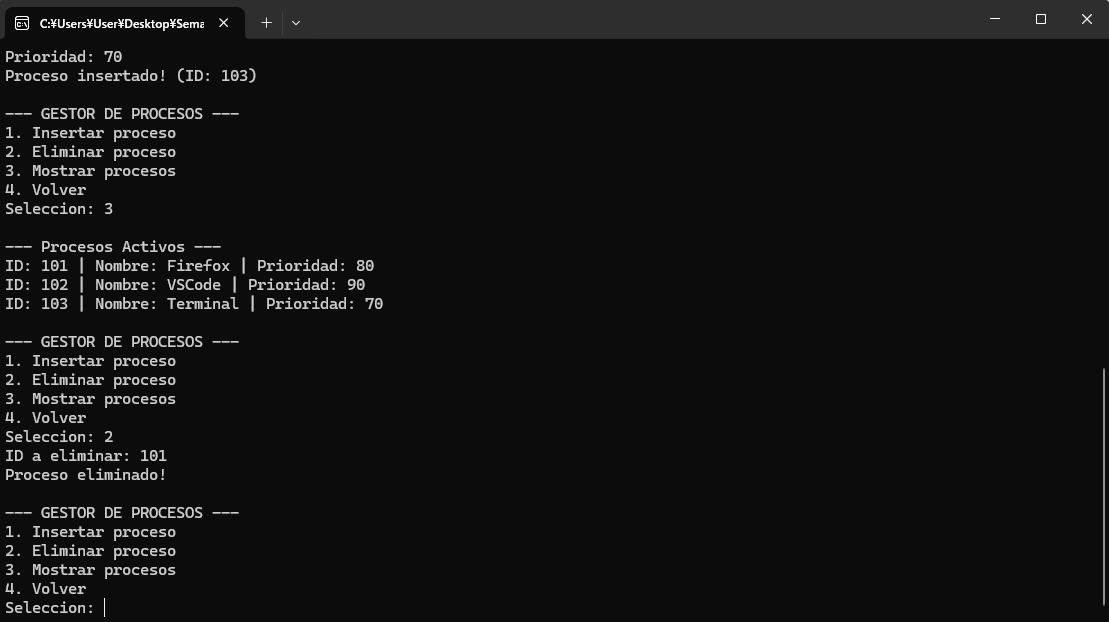


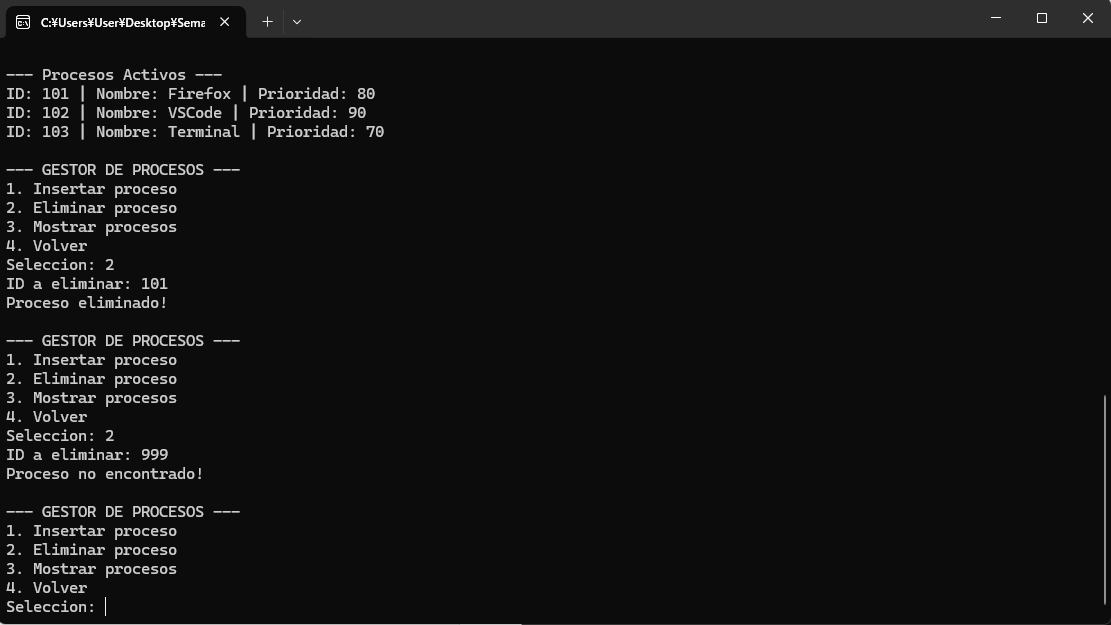




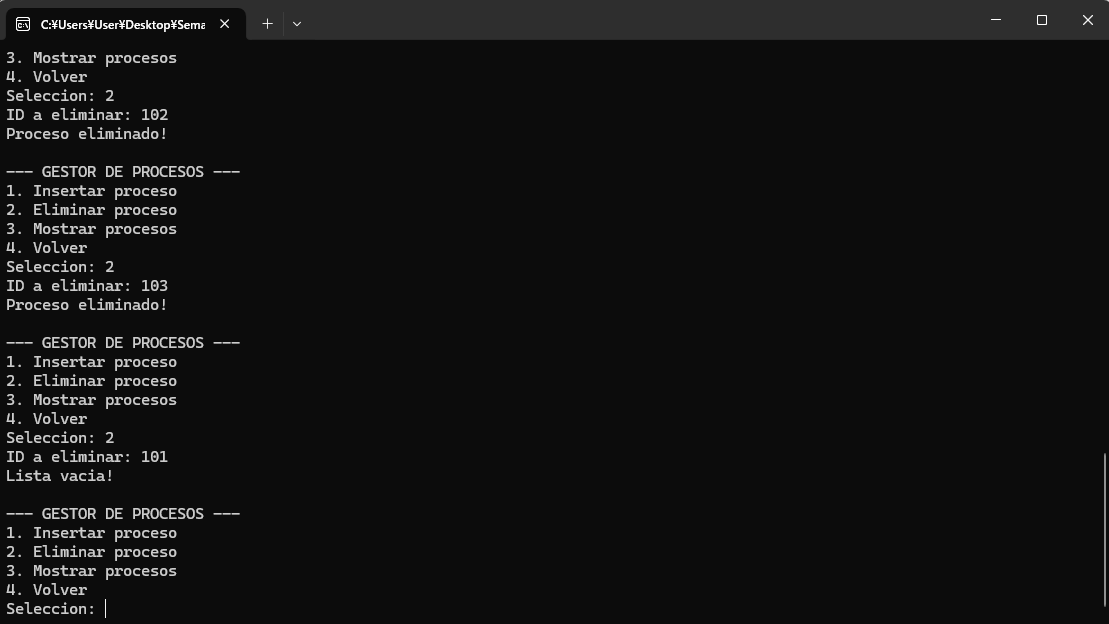
Inserción y mostrado de procesos:



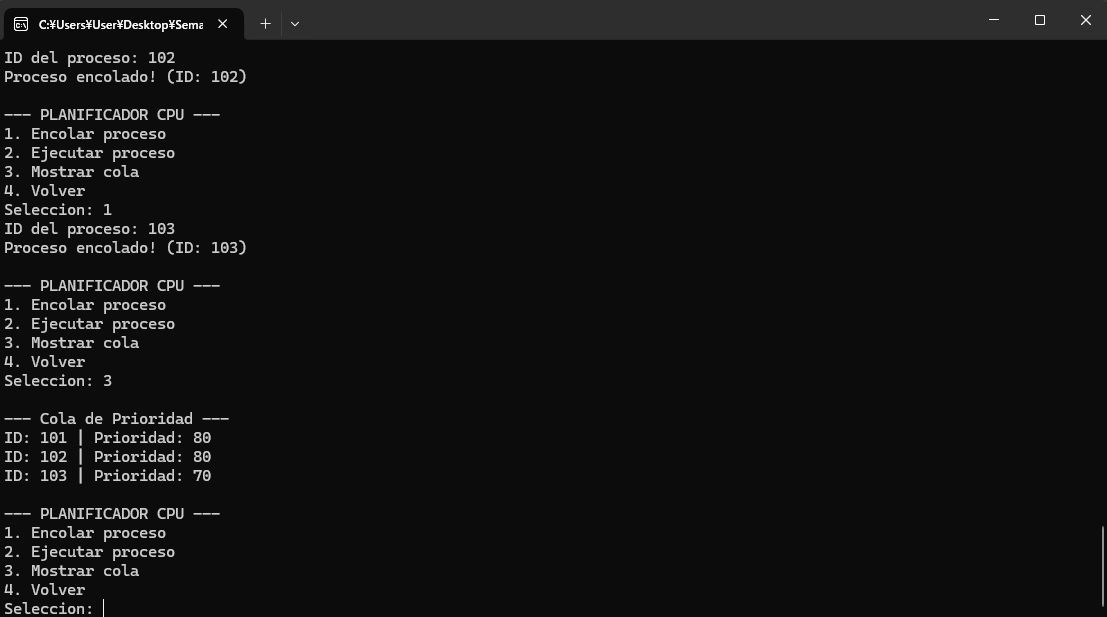
Eliminar proceso existente e inexistente:

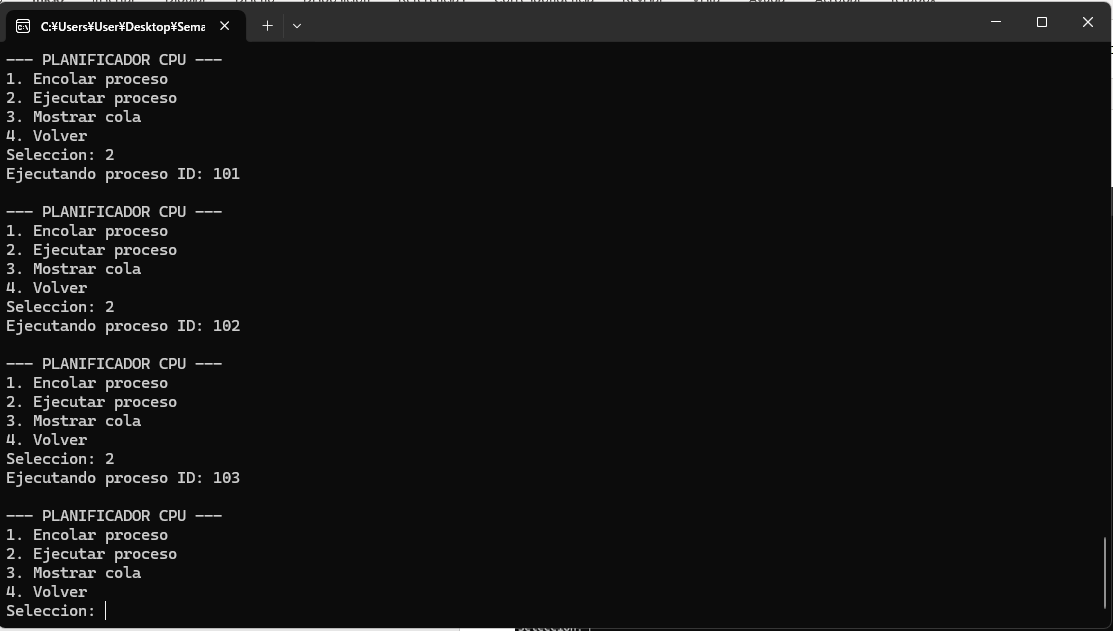


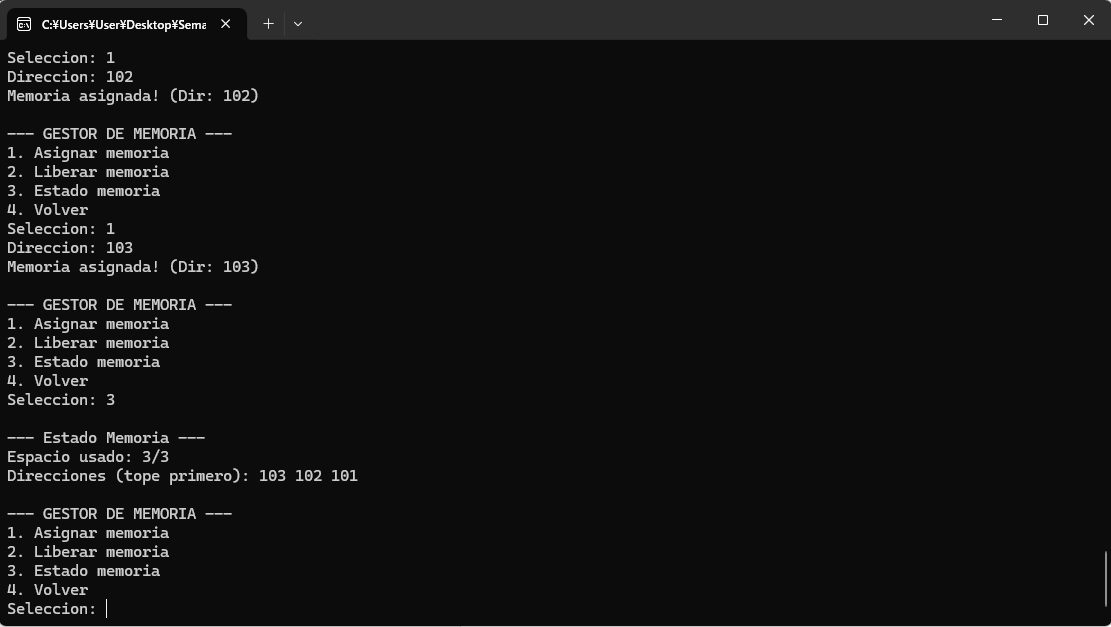
Eliminar lista vacía:



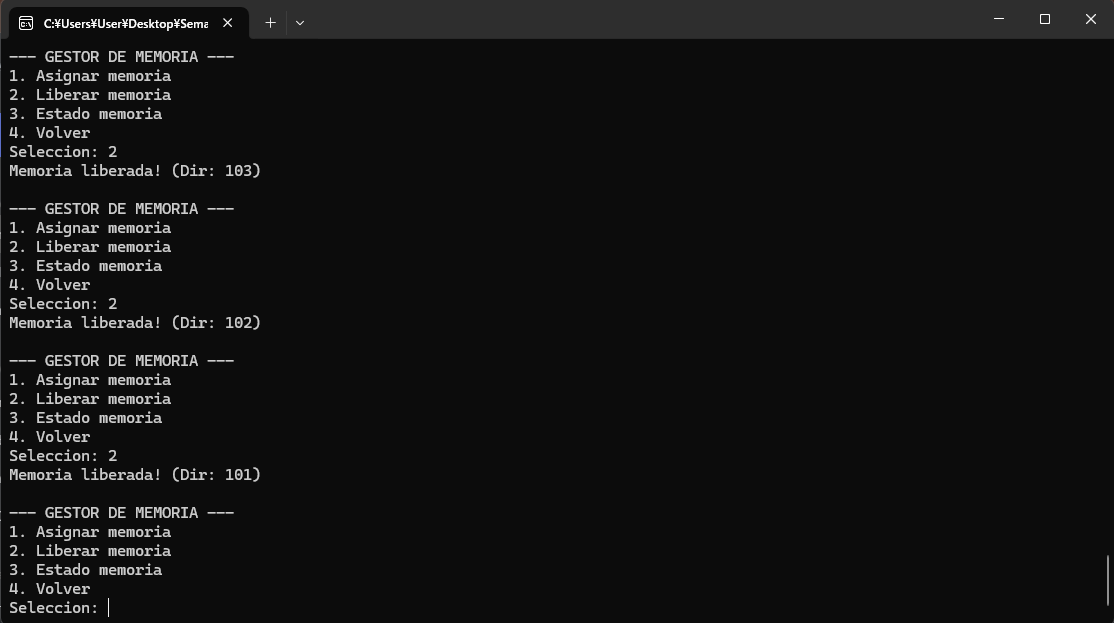
Encolar y mostrar procesos:



Ejecutar Procesos:

Asignación y mostrado de memoria:

Liberación de Memoria:



1. **GITHUB:**

**Link:** [**https://github.com/JBranez/Simulacion-de-Sistema-Operativo-Estructura-de-Datos-.git**](https://github.com/JBranez/Simulacion-de-Sistema-Operativo-Estructura-de-Datos-.git)

