GESTOR DE NOTAS ACADEMICAS

Manual tecnico

Ingeniería en sistemas, 2do semestre

Angel Marco Alberto Villagran Paredez

a. Descripción técnica general del sistema

Este documento describe la arquitectura y funcionalidad general del sistema de consola denominado "Gestor de Notas Académicas", implementado en Python. El sistema está diseñado para la administración básica de calificaciones de cursos, sirviendo como un ejercicio práctico para demostrar el uso de diversas estructuras de datos y algoritmos fundamentales.

Estructuras de Datos Principales

El sistema se basa en tres estructuras de datos fundamentales para gestionar la información y las operaciones:

- **cursos (Lista/Arreglo):** Es la estructura de datos principal. Almacena todos los cursos y sus respectivas notas como una **lista de tuplas** (nombre, nota).
- **cola_revision (Cola - *FIFO*):** Una lista utilizada para **simular una cola** de solicitudes de revisión de notas, procesando las solicitudes en el orden en que fueron ingresadas.
- **pila_historial (Pila - *LIFO*):** Una lista utilizada para **simular una pila** que registra los cambios importantes (actualizaciones y eliminaciones) en las notas, permitiendo ver el historial de manera inversa (el cambio más reciente primero).

---

2. Módulos y Funcionalidades Principales

El sistema se organiza en funciones modulares, cada una dedicada a una tarea específica, accesible a través de un menú principal.

2.1. Gestión de Datos (CRUD)

Función	**Descripción**
registrar_curso()	Agrega un nuevo curso con validación de nombre único y de nota (0-100).
actualizar_nota()	Modifica la nota de un curso existente y registra el cambio en la pila de historial.
eliminar_curso()	Remueve un curso existente tras confirmación y registra la eliminación en la pila de historial.

Función	Descripción
mostrar_cursos()	Lista todos los cursos y sus notas en el orden actual de registro.

B.) *Estructura general del código*

. Definición de Variables Globales (Estructuras de Datos)
Al inicio del script, se inicializan las estructuras de datos que actúan como la "memoria" del sistema. Estas son variables globales utilizadas por todas las funciones:

- **cursos = []**: La lista principal que almacena los datos de los cursos (tuplas (nombre, nota)).
- **cola_revision = []**: Lista utilizada como una estructura de **Cola** (FIFO).
- **pila_historial = []**: Lista utilizada como una estructura de **Pila** (LIFO) para registrar cambios.

2. Funciones de Interfaz y Utilidad
Estas funciones son esenciales para la interacción con el usuario y la validación de datos:

- **mostrar_menu()**: Imprime el menú de opciones en la consola.
- **validar_nota(nota_str)**: Función auxiliar que maneja la validación de entrada, asegurando que la nota sea un número decimal entre 0 y 100.

3. Funciones de Operaciones CRUD y Procesamiento
El núcleo del sistema, donde cada función maneja una opción del menú. Estas funciones implementan la lógica de negocio y manipulan la lista cursos:

- **Registro y Modificación (CRUD):**
  - registrar_curso()
  - actualizar_nota() (Incluye registro en la pila_historial)
  - eliminar_curso() (Incluye registro en la pila_historial)
- **Análisis y Reporte:**
  - mostrar_cursos()
  - calcular_promedio()
  - contar_aprobados_reprobados()

4. Funciones de Algoritmos Específicos
Estas funciones están dedicadas a demostrar la implementación de algoritmos de búsqueda, ordenamiento y el uso explícito de las estructuras de pila y cola:

- **Algoritmos de Búsqueda:**
  - buscar_curso_lineal()
  - buscar_curso_binaria()
- **Algoritmos de Ordenamiento (sobre una copia de cursos):**
  - ordenar_por_nota_burbuja() (Algoritmo **Burbuja**)

    - ordenar_por_nombre_insercion() (Algoritmo **Inserción**)
- **Demostración de Estructuras:**
    - simular_cola_revision() (Uso de la **Cola**)
    - mostrar_historial_cambios() (Uso de la **Pila**)

5. Bloque de Ejecución Principal
Esta sección controla el flujo del programa:

- **main()**: La función principal que contiene el bucle infinito (while True) del programa, gestionando la entrada del usuario (input) y llamando a la función correspondiente según la opción seleccionada.
- **if __name__ == "__main__":**: El estándar de Python para iniciar la ejecución del programa llamando a la función main().

C) Explicación del uso de listas, pilas, colas, etc.

La lista cursos es la estructura de datos principal que almacena el conjunto de datos del sistema.

Estructura	Implementación en Código	Lógica de Uso
Lista/Arreglo Dinámico	cursos = []	Se usa como un almacén flexible de registros, donde cada elemento es una tupla (nombre, nota). Permite: - Acceso por Índice: para actualizar o eliminar un elemento en una posición conocida. - Iteración: para calcular promedios, mostrar todos los elementos o realizar búsquedas lineales. - Implementación de Algoritmos: Es la base para los métodos de Búsqueda Lineal, Búsqueda Binaria, Ordenamiento Burbuja y Ordenamiento por Inserción.

---

2. Pilas (pila_historial)

La pila (Stack) es una estructura de datos que sigue el principio LIFO (Last-In, First-Out, el último en entrar es el primero en salir). Se usa para llevar un registro temporal de acciones recientes.

Estructura	Implementación en Código	Lógica de Uso
Pila (LIFO)	pila_historial = []	Entrada (Push): Se usa el método .append() para agregar un nuevo registro (el cambio más reciente) al final de la lista.
		Salida (Pop): Aunque el código solo la muestra (iterando desde el final con reversed()), una operación típica de pila sería .pop() para deshacer la última acción registrada. Esto permite ver el historial de cambios (actualizaciones y eliminaciones) en orden cronológico inverso (lo más nuevo primero).

---

3. Colas (cola_revision)

La cola (Queue) es una estructura de datos que sigue el principio FIFO (First-In, First-Out, el primero en entrar es el primero en salir). Se usa para manejar solicitudes de forma secuencial.

Estructura	Implementación en Código	Lógica de Uso
Cola (FIFO)	cola_revision = []	Entrada (Enqueue): Se usa el método .append() para agregar una nueva solicitud de revisión al final de la cola.
		Salida (Dequeue): El código simula el "procesamiento" iterando sobre la lista en su orden original, demostrando que la solicitud que entró primero es la que se atiende primero. En una implementación estricta, se usaría .pop(0) para remover el primer elemento y mantener el orden FIFO.

d) Justificación de los algoritmos de ordenamiento implementados

Justificación	Detalles de la Implementación
Utilidad en Ordenamiento de Cadenas	Se usa para ordenar los cursos alfabéticamente por nombre. Es particularmente eficiente para listas que ya están parcialmente ordenadas o para conjuntos de datos pequeños.
Preparación para Búsqueda Binaria	Es crucial para la función buscar_curso_binaria. La búsqueda binaria requiere que los datos estén ordenados para funcionar correctamente. Ordenar los cursos por nombre mediante Inserción es el paso lógico previo para poder aplicar la búsqueda binaria sobre ese mismo criterio.
Demostración de un Enfoque Diferente	A diferencia del Burbuja (que hace intercambios), el Inserción es un algoritmo de desplazamiento, donde cada elemento se inserta en su posición correcta dentro de la sublista ya ordenada. Esto ofrece una perspectiva diferente en la enseñanza de los métodos de ordenamiento.

e) Documentación breve de cada función o módulo

Función	Estructura/Algoritmo	Descripción Breve
buscar_curso_lineal()	Búsqueda Lineal	Busca un curso por nombre iterando secuencialmente sobre toda la lista.
buscar_curso_binaria()	Búsqueda Binaria	Busca un curso por nombre de forma eficiente (dividir y conquistar), pero requiere que la lista esté ordenada por nombre.
ordenar_por_nota_burbuja()	Ordenamiento Burbuja	Ordena una copia de los cursos por nota de forma descendente mediante comparaciones e intercambios.
ordenar_por_nombre_insercion()	Ordenamiento por Inserción	Ordena una copia de los cursos alfabéticamente por nombre insertando cada elemento en su lugar correcto.
simular_cola_revision()	Cola (FIFO)	Permite ingresar solicitudes de revisión que son "procesadas" en orden de primero en llegar, primero en ser atendido.
mostrar_historial_cambios()	Pila (LIFO)	Muestra los registros de cambios recientes (actualizaciones/eliminaciones) en orden inverso (el cambio más reciente aparece primero).

f). Pseudocodigo principal

```
INICIO PROGRAMA

// 1. Inicialización de Estructuras Globales

VARIABLES GLOBALES:

    cursos = []         // Lista principal de (nombre, nota)

    cola_revision = []  // Lista para la simulación FIFO
```

```
    pila_historial = [] // Lista para la simulación LIFO

// 2. Bucle Principal de Interfaz

MIENTRAS VERDADERO HACER:

    // a. Mostrar Menú

    LLAMAR FUNCION mostrar_menu()

    LEER opcion_seleccionada


    // b. Procesar Opciones (Selección Múltiple)

    SEGUN opcion_seleccionada:

        CASO '1': LLAMAR FUNCION registrar_curso()

        CASO '2': LLAMAR FUNCION mostrar_cursos()

        CASO '3': LLAMAR FUNCION calcular_promedio()

        CASO '4': LLAMAR FUNCION contar_aprobados_reprobados()

        CASO '5': LLAMAR FUNCION buscar_curso_lineal()

        CASO '6': LLAMAR FUNCION actualizar_nota()

        CASO '7': LLAMAR FUNCION eliminar_curso()

        CASO '8': LLAMAR FUNCION ordenar_por_nota_burbuja()

        CASO '9': LLAMAR FUNCION ordenar_por_nombre_insercion()

        CASO '10': LLAMAR FUNCION buscar_curso_binaria()

        CASO '11': LLAMAR FUNCION simular_cola_revision()

        CASO '12': LLAMAR FUNCION mostrar_historial_cambios()


        CASO '13':

            IMPRIMIR "Saliendo del sistema..."

            ROMPER BUCL

        DEFECTO:

            IMPRIMIR "Opción no válida. Intente de nuevo."
```

```
    // c. Espera para Continuar
    ESPERAR TECLA ENTER
FIN MIENTRAS
FIN PROGRAMA
```