Universidad Mariano Gálvez de Guatemala
Facultad de Ingeniería en Sistemas de Información y Ciencias de la Computación
Curso: Algoritmos
Catedrático: Miguel Alejandro Catalán López

Manual Técnico

Nombre: Vivian Marina Guadalupe
Apellidos: Cordova Figueroa
Carné: 7590-25-11480

Gestor de notas académico **(Manual Técnico)**

El **Gestor de Notas Académicas** es un programa desarrollado en **Python** que permite administrar cursos y sus respectivas calificaciones mediante un menú interactivo.
El sistema fue diseñado con el propósito de aplicar estructuras de datos (listas, pilas, colas y diccionarios) y algoritmos de búsqueda y ordenamiento, de forma práctica y didáctica.

Funcionalidades principales:

- Registro, modificación y eliminación de cursos y notas.
- Cálculo de promedios, conteo de aprobados y reprobados.
- Búsqueda de cursos por nombre (búsqueda lineal y binaria).
- Ordenamiento de cursos por nombre o nota.
- Simulación de una cola de solicitudes de revisión (FIFO).
- Visualización de un historial de cambios usando una pila (LIFO).

El sistema funciona completamente en consola y no requiere librerías externas, utilizando exclusivamente elementos nativos de Python. Esto significa que se ejecuta correctamente en cualquier entorno donde esté instalada una versión estándar de **Python (≥3.6)**, sin necesidad de hacer instalaciones con pip.

Estructura general del código
El código está organizado de forma modular, dividiendo las operaciones en funciones específicas.
La estructura general se puede representar así:

Gestor de Notas Académicas
│
├── Variables globales
│ └── historial_cambios → lista que actúa como pila (LIFO)
│
├── Funciones principales
│ ├── registrar_curso() → Añadir curso y nota
│ ├── mostrar_cursos() → Mostrar cursos y calificaciones
│ ├── calcular_promedio() → Calcular promedio general
│ ├── contar_aprobados_reprobados() → Contar según nota mínima
│ ├── buscar_curso_lineal() → Búsqueda simple de curso

```
│   ├── actualizar_nota() → Modificar nota y registrar cambio
│   ├── eliminar_curso() → Eliminar curso y registrar cambio
│   ├── ordenar_por_nota() → Ordenamiento burbuja (descendente)
│   ├── ordenar_por_nombre() → Ordenamiento por inserción (alfabético)
│   ├── buscar_curso_binaria() → Búsqueda binaria por nombre
│   ├── simular_cola_revision() → Simulación de cola FIFO
│   └── mostrar_historial_cambios() → Mostrar pila de cambios (LIFO)
│
└── main() → Controlador del menú principal del programa
```

Cada módulo está diseñado para realizar una tarea específica, facilitando el mantenimiento, la legibilidad y la reutilización del código.

Explicación del uso de estructuras de datos

1. Diccionario (dict)

- **Uso:** Para almacenar los cursos y sus notas.

```
cursos = {"Matemáticas": 85, "Física": 90}
```

Ventajas:

- Permite búsqueda rápida por nombre de curso.
- Permite modificar o eliminar elementos fácilmente.

2. Lista como pila (list simulando LIFO)

- **Variable:** historial_cambios
- **Uso:** Almacena los cambios realizados (actualizaciones o eliminaciones de cursos).

```
historial_cambios.append("Se actualizó: Matemáticas - Nota anterior: 80 → Nueva: 90")
```

Comportamiento:

- Último en entrar → primero en mostrarse.
- Se simula una **pila (LIFO: Last In, First Out)**.

Ejemplo práctico: Permite revisar los últimos cambios realizados en el sistema.

3. Lista como cola (list simulando FIFO)

- **Uso en:** simular_cola_revision()
  cola_solicitudes.append("Matemáticas") # Enqueue
  curso_a_revisar = cola_solicitudes.pop(0) # Dequeue
- **Comportamiento:**
  - Primero en entrar → primero en salir (**FIFO**).
- **Ejemplo práctico:** Simula el orden en que los estudiantes solicitan revisión de cursos.

4. Listas para ordenamiento y búsqueda

- Las funciones de ordenamiento convierten el diccionario cursos a una lista de tuplas (curso, nota) para manipular los datos y aplicar algoritmos de ordenamiento personalizados.

Justificación de los algoritmos de ordenamiento

El código implementa dos algoritmos clásicos de ordenamiento:

1. Ordenamiento Burbuja (Bubble Sort) – por Nota

- Implementado en ordenar_por_nota().
- **Motivo de uso:**
  Es un algoritmo simple y didáctico, ideal para enseñar la comparación e intercambio de elementos.
  Aunque no es el más eficiente, es excelente para listas pequeñas.
- **Funcionamiento:**
  Recorre la lista varias veces y va "empujando" el elemento mayor hacia el final (o el menor, según el orden).

2. Ordenamiento por Inserción (Insertion Sort) – por Nombre

- Implementado en ordenar_por_nombre().
- **Motivo de uso:**
  Es intuitivo y eficiente para conjuntos pequeños de datos, además de mantener la estabilidad del orden (no altera el orden relativo de elementos con igual clave).
- **Funcionamiento:**
  Inserta cada elemento en su posición correcta en una lista ya parcialmente ordenada.

Documentación breve de las funciones

Función	Descripción	Tipo de estructura o algoritmo usado
registrar_curso(cursos)	Solicita nombre y nota del curso y lo guarda en el diccionario.	Diccionario
mostrar_cursos(cursos)	Muestra todos los cursos registrados.	Iteración
calcular_promedio(cursos)	Calcula el promedio de todas las notas registradas.	Operaciones matemáticas
contar_aprobados_reprobados(cursos)	Cuenta cuántos cursos están aprobados o reprobados (≥60).	Estructura condicional
buscar_curso_lineal(cursos)	Busca un curso por nombre usando búsqueda lineal (coincidencia parcial).	Búsqueda lineal
actualizar_nota(cursos)	Permite cambiar la nota de un curso y registra el cambio en la pila.	Diccionario + Pila
eliminar_curso(cursos)	Elimina un curso y registra la acción en la pila.	Diccionario + Pila
ordenar_por_nota(cursos)	Ordena los cursos por nota descendente usando burbuja.	Bubble Sort
ordenar_por_nombre(cursos)	Ordena los cursos por nombre alfabéticamente.	Insertion Sort
buscar_curso_binaria(cursos)	Busca un curso por nombre usando búsqueda binaria (requiere lista ordenada).	Binary Search
simular_cola_revision()	Simula una cola FIFO de solicitudes de revisión.	Cola (FIFO)
mostrar_historial_cambios()	Muestra los cambios recientes en orden inverso (últimos primero).	Pila (LIFO)
main()	Controla el flujo del programa mediante un menú interactivo.	Control general

Conclusión técnica

El sistema **Gestor de Notas Académicas** combina estructuras de datos fundamentales (listas, pilas, colas y diccionarios) con algoritmos clásicos de ordenamiento y búsqueda.
Su diseño modular permite comprender el comportamiento de cada estructura y su aplicación práctica en la gestión de información.
Además, el uso de una **pila de historial** y una **cola de solicitudes** simula procesos reales de registro y atención secuencial, reforzando los principios de programación estructurada.