Manual Técnico

Nombre del estudiante: Jorge Antonio González Valladares

Nombre del proyecto: Gestor de Notas Académicas

Descripción técnica general del sistema

El sistema 'Gestor de Notas Académicas' está desarrollado en Python y permite administrar cursos y sus calificaciones. Utiliza estructuras de datos como listas, pilas y colas para manejar información dinámica. Incluye funciones para registrar, actualizar, eliminar, ordenar y buscar cursos, así como simular una cola de solicitudes de revisión.

Estructura general del código

El programa está estructurado de manera modular, con funciones independientes para cada operación. Cuenta con un menú principal que dirige las opciones y un bucle principal que mantiene la ejecución hasta que el usuario decida salir.

Uso de estructuras de datos

- Listas: Se utilizan para almacenar los cursos y sus respectivas notas.
- Cola: Implementada con una lista para manejar solicitudes de revisión de notas.
- Pila: Implementada con una lista para registrar el historial de cambios realizados en las notas.

Justificación de los algoritmos de ordenamiento

Se implementa el algoritmo de ordenamiento burbuja (Bubble Sort) para ordenar tanto por nombre como por nota. Este algoritmo fue elegido por su simplicidad y adecuación a conjuntos de datos pequeños.

Se implemento también el algortimo de ordenamiento por insercion debico a:

El ordenamiento por inserción recorre la lista y va insertando cada elemento en la posición correcta, como cuando ordenás cartas en la mano 🃏.

Es más eficiente que el burbuja en listas pequeñas o casi ordenadas.

Mantiene la asociación entre curso y nota, ya que mueve ambos al mismo tiempo.

Documentación breve de cada función o módulo

Cada función realiza una tarea específica dentro del sistema:

- registrar_curso_lista(): Registra nuevos cursos y notas.

- mostrar_cursos_lista(): Muestra los cursos y notas registrados.

- promedio(): Calcula el promedio general de notas.

- cursos_aprobados_reprobados(): Clasifica los cursos aprobados y reprobados.

- buscar_curso(): Realiza búsqueda lineal.

- actualizar_nota(): Actualiza notas existentes y registra cambios.

- borrar_curso_lista(): Elimina cursos registrados.

- ordenar_por_nota_burbuja(): Ordena los cursos por nota.

- ordenar_por_nombre(): Ordena los cursos alfabéticamente.

- buscar_curso_binario(): Realiza búsqueda binaria sobre una lista ordenada.

- simular_cola_revision(): Gestiona solicitudes de revisión de notas.

- mostrar_historial(): Muestra los cambios almacenados en la pila.

Diagrama general del sistema o pseudocódigo principal

INICIO

```
  // ==================== DECLARACIÓN DE ESTRUCTURAS ====================

  cursos_lista ← ["Mate Discreta", "Contabilidad 2", "Pre Cálculo", "Algebra Lineal",
"Algoritmos"]
```

```
notas_lista ← [85, 65, 61, 84, 99]
cola_revisiones ← []            // Simula una cola de solicitudes
historial_cambios ← []          // Simula una pila de cambios realizados

// ==================== DEFINICIÓN DE FUNCIONES ====================

FUNCION mostrar_menu()
   IMPRIMIR el menú principal con las 13 opciones
FIN FUNCION

FUNCION registrar_curso_lista()
   LEER cantidad de cursos a registrar
   PARA i DESDE 1 HASTA n HACER
      LEER nombre del curso
      MIENTRAS el nombre esté vacío O ya exista en cursos_lista HACER
         PEDIR nuevamente el nombre
      FIN MIENTRAS

      LEER nota del curso
      MIENTRAS nota < 0 O nota > 100 HACER
         PEDIR nuevamente la nota
      FIN MIENTRAS

      AGREGAR curso a cursos_lista
      AGREGAR nota a notas_lista
   FIN PARA
```

```
    MOSTRAR todos los cursos registrados
FIN FUNCION

FUNCION mostrar_cursos_lista()
    SI cursos_lista ESTÁ VACÍA ENTONCES
        IMPRIMIR "No hay cursos registrados"
    SINO
        PARA i DESDE 0 HASTA longitud(cursos_lista) - 1 HACER
            IMPRIMIR curso[i], nota[i]
        FIN PARA
    FIN SI
FIN FUNCION

FUNCION promedio()
    SI cursos_lista ESTÁ VACÍA ENTONCES
        IMPRIMIR "No hay cursos registrados"
    SINO
        suma ← SUMA(notas_lista)
        promedio ← suma / longitud(notas_lista)
        IMPRIMIR promedio
    FIN SI
FIN FUNCION

FUNCION cursos_aprobados_reprobados()
    aprobados ← []
    reprobados ← []
```

```
    PARA i DESDE 0 HASTA longitud(cursos_lista) - 1 HACER
      SI notas_lista[i] >= 61 ENTONCES
        AGREGAR (curso, nota) A aprobados
      SINO
        AGREGAR (curso, nota) A reprobados
      FIN SI
    FIN PARA
    IMPRIMIR aprobados y reprobados
FIN FUNCION

FUNCION buscar_curso()
    LEER nombre del curso
    SI nombre EXISTE EN cursos_lista ENTONCES
      índice ← posición de nombre en cursos_lista
      nota ← notas_lista[índice]
      IMPRIMIR curso, nota, y estado (Aprobado o Reprobado)
    SINO
      IMPRIMIR "Curso no encontrado"
    FIN SI
FIN FUNCION

FUNCION actualizar_nota()
    LEER nombre del curso
    SI nombre EXISTE EN cursos_lista ENTONCES
      índice ← posición de nombre
      LEER nueva_nota
```

```
      notas_lista[índice] ← nueva_nota
      AGREGAR cambio a historial_cambios
    SINO
      IMPRIMIR "Curso no encontrado"
    FIN SI
FIN FUNCION

FUNCION borrar_curso_lista()
    MOSTRAR cursos registrados con índice
    LEER número de curso a eliminar
    SI número válido ENTONCES
      CONFIRMAR eliminación
      ELIMINAR curso y nota de ambas listas
    SINO
      IMPRIMIR "Número inválido"
    FIN SI
FIN FUNCION

FUNCION ordenar_por_nota_burbuja()
    PARA i DESDE 0 HASTA n-1 HACER
      PARA j DESDE 0 HASTA n-i-2 HACER
        SI notas_lista[j] > notas_lista[j+1] ENTONCES
          INTERCAMBIAR notas_lista[j] CON notas_lista[j+1]
          INTERCAMBIAR cursos_lista[j] CON cursos_lista[j+1]
        FIN SI
      FIN PARA
```

```
    FIN PARA
    IMPRIMIR cursos ordenados por nota
FIN FUNCION

FUNCION ordenar_por_nombre_insercion()
    PARA i DESDE 1 HASTA longitud(cursos_lista)-1 HACER
        curso_actual ← cursos_lista[i]
        nota_actual ← notas_lista[i]
        j ← i - 1
        MIENTRAS j >= 0 Y cursos_lista[j] > curso_actual HACER
            cursos_lista[j + 1] ← cursos_lista[j]
            notas_lista[j + 1] ← notas_lista[j]
            j ← j - 1
        FIN MIENTRAS
        cursos_lista[j + 1] ← curso_actual
        notas_lista[j + 1] ← nota_actual
    FIN PARA
    IMPRIMIR cursos ordenados alfabéticamente
FIN FUNCION

FUNCION buscar_curso_binario()
    ORDENAR cursos por nombre
    izq ← 0
    der ← longitud(cursos_lista) - 1
    LEER nombre_buscado
    MIENTRAS izq ≤ der HACER
```

```
      medio ← (izq + der) DIV 2
      SI cursos_lista[medio] = nombre_buscado ENTONCES
        IMPRIMIR curso y nota
        SALIR
      SINO SI nombre_buscado < cursos_lista[medio] ENTONCES
        der ← medio - 1
      SINO
        izq ← medio + 1
      FIN SI
    FIN MIENTRAS
    IMPRIMIR "Curso no encontrado"
FIN FUNCION

FUNCION simular_cola_revision()
    MIENTRAS VERDADERO HACER
      IMPRIMIR submenú de revisiones
      LEER opción
      SEGÚN opción HACER
        CASO 1:
          LEER nombre del curso
          AGREGAR curso a cola_revisiones
        CASO 2:
          SI cola NO vacía ENTONCES
            curso ← cola_revisiones[0]
            LEER nueva nota
            ACTUALIZAR nota
```

```
            AGREGAR cambio al historial
          SINO
            IMPRIMIR "Cola vacía"
          FIN SI
        CASO 3:
          MOSTRAR solicitudes pendientes
        CASO 4:
          SALIR DEL SUBMENÚ
      FIN SEGÚN
    FIN MIENTRAS
FIN FUNCION

FUNCION mostrar_historial()
    SI historial_cambios NO está vacío ENTONCES
      MOSTRAR cada registro en orden inverso
    SINO
      IMPRIMIR "No hay cambios registrados"
    FIN SI
FIN FUNCION

FUNCION ejecutar_opcion(opcion)
    SEGÚN opcion HACER
      CASO 1: registrar_curso_lista()
      CASO 2: mostrar_cursos_lista()
      CASO 3: promedio()
      CASO 4: cursos_aprobados_reprobados()
```

```
        CASO 5: buscar_curso()
        CASO 6: actualizar_nota()
        CASO 7: borrar_curso_lista()
        CASO 8: ordenar_por_nota_burbuja()
        CASO 9: ordenar_por_nombre_insercion()
        CASO 10: buscar_curso_binario()
        CASO 11: simular_cola_revision()
        CASO 12: mostrar_historial()
        CASO 13: SALIR DEL PROGRAMA
        OTRO: IMPRIMIR "Opción inválida"
    FIN SEGÚN
  FIN FUNCION

  // ==================== PROGRAMA PRINCIPAL ====================
  MIENTRAS VERDADERO HACER
    mostrar_menu()
    LEER opción
    ejecutar_opcion(opción)
    PREGUNTAR si desea realizar otra operación
    SI respuesta ≠ "s" ENTONCES
      IMPRIMIR "Programa terminado"
      SALIR DEL BUCLE
    FIN SI
  FIN MIENTRAS

FIN
```