Universidad Mariano Gálvez

Facultad: Ingeniería en Sistemas de Información y Ciencias de Computación

Curso: Algoritmos

Docente: Ing. Miguel Catalán

Manual Técnico

Gestor de Notas

Nombre: Wilmer Missael González Oroxón

Carnet: 7590-25-282

Sección: A

Fecha: 24/10/2025

Gestor de Notas

El Gestor de Notas, es un proyecto de Python que permite llevar un control básico sobre cursos académicos y notas. Permite agregar nuevos registros cursos, la edición, eliminación, ordenamientos de 2 formas, por notas y por nombres, incluyendo una simulación de revisión, todo a través de una línea de comandos. Todos los datos están guardados en una sola lista, también se utiliza una lista llamada historial para guardar cambios.

Estructura General

El proyecto consta de un solo archivo Proyecto_gestor_de_notas.py, la cual está estructurado de la siguiente manera:

- Declaración de variables globales: la lista de los cursos y el historial.
- Definición de Funciones y procedimientos que se agrupan de la siguiente manera:
    - Registro y gestión de cursos: nuevo_curso(), mostrar_lista().
    - Estadística: promedio_general(), cursos_aprobados().
    - Búsqueda: buscar_curso(), mostrar_busqueda(), búsqueda_binaria por_nombre.
    - Modificación de cursos: actualizar_nota(), eliminar_curso().
    - Ordenamiento: ordenamiento_por_nota(), ordenamiento_por_nombre().
    - Simulación de solicitudes: solicitud_de revision().
    - Historial de cambios: mostrar_historial().
- Declaración de una variable string que contiene el menú para mostrar al usuario.
- Bucle principal del programa que se encarga solicitar la opción que desea hacer el programa, valida si hay cursos, llama a las funciones o procedimientos, y controla las repeticiones y salidas.

Uso de listas pilas colas

Listas: Es principalmente utilizada de manera global para guardar los cursos, y cada elemento de esta lista incluye: nombre del curso, nota, y estado (aprobado o reprobado), también se utiliza una lista para guardar el historial de cambios cuando se modifica una nota o se elimina algún curso, además una lista es utilizada en algunas funciones en donde es necesario buscar algún curso(ej. buscar_curso()), la lista se usa para que almacene los cursos que coincidan con la búsqueda.

Pilas: Pila es utilizada en la función de mostrar historial por que almacena los cambios hechos, y muestra de primero el ultimo cambio realizado.

Cola: Es utilizada en la simulación de revisión de solicitudes, conforme se ingrese el nombre de los cursos se estará revisando, el primero que se escribe, es el primero que se revisa.

Justificación de los algoritmos de ordenamiento

Ordenamiento por burbuja: Se utiliza en el ordenamiento por nota, la cual ordena las los cursos por nota de mayor a menor, este algoritmo trabajo muy bien con listas pequeñas, y siendo este un gestor de notas, se espera que no contenga una lista grande.

Ordenamiento por inserción: Es utilizado en el ordenamiento por nombres, ordenándolos alfabéticamente, este algoritmo se ha implementado porque es muy adecuado para lista casi ordenadas y pequeñas,

Documentación de cada función y procedimiento

nuevo_curso(nombre, nota) : Esta función recibe 2 parámetros que son el nombre y la nota, y evalúa la nota si es mayor o igual a 60 para crear el dato de aprobado o reprobado, y retorna una pequeña lista de tres elementos, nombre del curso, nota, aprobado.

mostrar_lista(lista): este procedimiento recibe como parámetro una lista(en este gestor se utilizan listas que en cada elemento incluyen, nombre del curso, nota, estado) la cual se recorre para desempaquetar los elementos y luego imprime el nombre del curso, nota y estado.

promedio_general(lista): Esta función recorreá la lista global de cursos y suma cada una de las notas, para luego dividirlas en el número total de cursos registrados y también redondear el resultado a dos decimales.

cursos_aprobados(lista): Hace el recorrido de la lista de los cursos, para contar los estados aprobados y reprobados y retornar el conteo en dos variables.

buscar_curso(lista, búsqueda): recibe un dato a buscar que se normaliza a minúscula, y la lista de cursos, esta función es un algoritmo de búsqueda lineal, en donde recorre cada elemento y compara el dato a buscar, para luego añadir las coincidencias en una lista llamada encontrados. La función permite realizar búsquedas parciales, y al final retorna la lista de encontrados.

mostrar_busqueda(lista, búsqueda): este procedimiento completa la función buscar curso porque recibe de igual manera el dato a buscar y una lista, pero la diferencia es que la lista que recibe la lista de encontrados, y luego imprime cada uno de los cursos, o bien, si la lista está vacía mostrará un mensaje indicando que no se encontraron coincidencias.

actualizar_nota(lista, nueva_nota): en esta función se asigna la nueva nota al índice 1 del curso(es donde esta la nota), y además también se modifica el estado (aprobado o reprobado) y al final guarda el cambio en la lista del historial.

eliminar_curso(lista_cursos, búsqueda): esta función permite eliminar un curso si el usuario lo confirma. Primero realiza una búsqueda lineal para poder encontrar coincidencias y si hay mas de una, la eliminación se hará del primer resultado. También se registrará la eliminación

en el historial. En consola mostrará los resultados de la búsqueda y pedirá confirmación al usuario para la eliminación, y al final muestra el resultado del proceso.

ordenamiento_por_nota(lista): en este ordenamiento se utiliza el ordenamiento por burbuja, y va comparando la nota de cada uno de los cursos para ordenarlos de mayor a menor.

ordenamiento_por_nombre(lista): ordena todos los cursos por nombre (en orden alfabético), se utiliza el ordenamiento por inserción.

busqueda_binario_por_nombre(lista,busqueda): esta función recibe como parámetro el dato o elemento a buscar y la lista, utiliza el algoritmo de busqueda binaria, en la que en cada comparación divide la lista en la mitad hasta llegar a la coincidencia.

solicitud_de_revision(): esta función pide al usuario nombre de cursos y los va almacenando en una cola (lista llamada solicitudes). Y termina de pedirlos cuando el usuario escriba fin, luego imprime las solicitudes conforme fueron añadidas. Y si no hay solicitudes, muestra un mensaje indicándolo. Esta es únicamente una simulación.

mostrar_historial(): imprime cada cambio realizado de actualización de nota o eliminación de curso, mostrando una pila, ultimo que entro es el primero que muestra. Si no hay cambios muestra un mensaje indicándolo.

Pseudocódigo

```
INICIO
  DEFINIR historial
  DEFINIR lista_cursos

  FUNCION nuevo_curso(nombre, nota)
    SI nota >= 60 ENTONCES
      aprobado = "Aprobado"
    SINO
      aprobado = "Reprobado"
    FIN SI
    RETORNAR [nombre, nota, aprobado]
  FIN FUNCION


  PROCEDIMIENTO mostrar_lista(lista)
    PARA i DESDE 1 HASTA longitud(lista) HACER
      nombre = lista[i][0]
      nota = lista[i][1]
```

```
        aprobado = lista[i][2]
        IMPRIMIR i, "Curso:", nombre, "Nota:", nota, aprobado
    FIN PARA
FIN PROCEDIMIENTO


FUNCION promedio_general(lista)
    suma_notas = 0
    cantidad = 0
    PARA cada curso EN lista HACER
        nota = curso[1]
        suma_notas = suma_notas + nota
        cantidad = cantidad + 1
    FIN PARA

    resultado = suma_notas / cantidad
    promedio = REDONDEAR(resultado, 2)
    RETORNAR promedio
FIN FUNCION

FUNCION cursos_aprobados(lista)
    aprobado = 0
    reprobado = 0
    PARA cada curso EN lista HACER
        estado = curso[2]
        SI estado = "Aprobado" ENTONCES
            aprobado = aprobado + 1
        SINO
            reprobado = reprobado + 1
        FIN SI
    FIN PARA
    RETORNAR aprobado, reprobado
FIN FUNCION


FUNCION buscar_curso(lista, busqueda)
    busqueda = MINUSCULAS(busqueda)
    encontrados = LISTA VACÍA
    PARA cada curso EN lista HACER
        nombre = curso[0]
        SI busqueda ESTÁ EN MINUSCULAS(nombre) ENTONCES
```

```
            AÑADIR curso A encontrados
        FIN SI
    FIN PARA
    RETORNAR encontrados
FIN FUNCION

PROCEDIMIENTO mostrar_busqueda(lista, busqueda)
    SI longitud(lista) > 0 ENTONCES
        IMPRIMIR "Cursos encontrados con ", busqueda
        PARA cada curso EN lista HACER
            IMPRIMIR "Nombre:", curso[0], "Nota:", curso[1], curso[2]
        FIN PARA
    SINO
        IMPRIMIR "No se encontraron cursos con ", busqueda
    FIN SI
FIN PROCEDIMIENTO

PROCEDIMIENTO actualizar_nota(lista, nueva_nota)
    PARA cada curso EN lista HACER
        nombre = curso[0]
        nota_anterior = curso[1]
        curso[1] = nueva_nota
        SI nueva_nota >= 60 ENTONCES
            curso[2] = "Aprobado"
        SINO
            curso[2] = "Reprobado"
        FIN SI
        IMPRIMIR "Nota actualizada exitosamente"
        AÑADIR "Se actualizó: " + nombre + " - Nota anterior: " + nota_anterior + " -> Nueva
nota: " + nueva_nota A historial
    FIN PARA
FIN PROCEDIMIENTO


PROCEDIMIENTO eliminar_curso(lista_cursos, busqueda)
    busqueda = MINUSCULAS(busqueda)
    encontrados = buscar_curso(lista_cursos, busqueda)

    SI longitud(encontrados) = 0 ENTONCES
        IMPRIMIR "No se encontraron cursos con ", busqueda
        RETORNAR
```

```
    FIN SI

    curso_a_eliminar = encontrados[0]
    IMPRIMIR "Curso encontrado:", curso_a_eliminar[0]

    IMPRIMIR "¿Desea eliminar el curso? (sí/no)"
    LEER confirmacion
    SI confirmacion EN ["sí", "si", "s"] ENTONCES
        ELIMINAR curso_a_eliminar DE lista_cursos
        AÑADIR "Se eliminó: " + curso_a_eliminar[0] A historial
        IMPRIMIR "Curso eliminado exitosamente"
    SINO
        IMPRIMIR "Eliminación cancelada"
    FIN SI
FIN PROCEDIMIENTO


FUNCION ordenamiento_por_nota(lista)
    n = longitud(lista)
    changed = VERDADERO
    comps = 0
    swaps = 0

    MIENTRAS changed = VERDADERO HACER
        changed = FALSO
        PARA i DESDE 0 HASTA n - 2 HACER
            comps = comps + 1
            SI lista[i][1] < lista[i+1][1] ENTONCES
                INTERCAMBIAR lista[i] Y lista[i+1]
                swaps = swaps + 1
                changed = VERDADERO
            FIN SI
        FIN PARA
        n = n - 1
    FIN MIENTRAS
    RETORNAR comps, swaps, lista
FIN FUNCION


FUNCION ordenamiento_por_nombre(lista)
    comps = 0
```

```
    movimientos = 0
    PARA i DESDE 1 HASTA longitud(lista)-1 HACER
        curso = lista[i]
        clave = MINUSCULAS(curso[0])
        j = i - 1
        MIENTRAS j >= 0 Y MINUSCULAS(lista[j][0]) > clave HACER
            lista[j+1] = lista[j]
            movimientos= movimientos + 1
            j = j - 1
        FIN MIENTRAS
        lista[j+1] = curso
    FIN PARA
    RETORNAR comps, movimientos, lista
FIN FUNCION


FUNCION busqueda_por_nombre_binario(lista, busqueda)
    min = 0
    max = longitud(lista) - 1
    busqueda = MINUSCULAS(busqueda)

    MIENTRAS min <= max HACER
        medio = (min + max) DIV 2
        nombre_curso = MINUSCULAS(lista[medio][0])

        SI busqueda ESTÁ EN nombre_curso ENTONCES
            RETORNAR lista[medio]
        SINO SI busqueda < nombre_curso ENTONCES
            max = medio - 1
        SINO
            min = medio + 1
        FIN SI
    FIN MIENTRAS
    RETORNAR -1
FIN FUNCION


PROCEDIMIENTO solicitud_de_revision()
    solicitudes = LISTA VACÍA
    MIENTRAS VERDADERO HACER
        IMPRIMIR "Ingrese curso para revisión (fin para terminar):"
```

```
      LEER n
      SI n = "FIN" ENTONCES
        SALIR DEL BUCLE
      SINO
        AÑADIR n A solicitudes
      FIN SI
    FIN MIENTRAS

    SI longitud(solicitudes) = 0 ENTONCES
      IMPRIMIR "No hay solicitudes ingresadas."
    SINO
      IMPRIMIR "Procesando solicitudes:"
      PARA cada i EN solicitudes HACER
        IMPRIMIR "Revisando:", i
      FIN PARA
    FIN SI
FIN PROCEDIMIENTO

PROCEDIMIENTO mostrar_historial()
    SI longitud(historial) = 0 ENTONCES
      IMPRIMIR "No hay historial de cambios."
    SINO
      IMPRIMIR "Historial de cambios:"
      PARA cada cambio EN historial HACER
        IMPRIMIR cambio
      FIN PARA
    FIN SI
FIN PROCEDIMIENTO


MENÚ PRINCIPAL

MIENTRAS VERDADERO HACER
    IMPRIMIR "GESTOR DE NOTAS ACADÉMICAS “
    IMPRIMIR "1. Registrar nuevo curso"
    IMPRIMIR "2. Mostrar todos los cursos"
    IMPRIMIR "3. Calcular promedio general"
    IMPRIMIR "4. Contar cursos aprobados y reprobados"
    IMPRIMIR "5. Buscar curso por nombre"
    IMPRIMIR "6. Actualizar nota de curso"
    IMPRIMIR "7. Eliminar curso"
```

```
        IMPRIMIR "8. Ordenar cursos por nota"
        IMPRIMIR "9. Ordenar cursos por nombre"
        IMPRIMIR "10. Buscar curso (binaria)"
        IMPRIMIR "11. Solicitudes de revisión"
        IMPRIMIR "12. Mostrar historial"
        IMPRIMIR "13. Salir"
    IMPRIMIR “Seleccione una opcion: "
    LEER opcion

        SEGÚN opcion HACER
          CASO 1:
            IMPRIMIR "Registrar nuevo curso"
            IMPRIMIR “Ingrese el nombre del curso”
            LEER nombre
            MIENTRAS nombre = “ “ HACER
                    IMPRIMIR “Ingrese un nombre valido”
                    LEER nombre
            IMPRIMIR “Ingrese la nota del curso”
            LEER nota
            curso = nuevo_curso(nombre, nota)
            AÑADIR curso A lista_cursos
          CASO 2:
            SI lista_curso = vacia ENTONCES
                    IMPRIMIR “No hay cursos registrados”
            mostrar_lista(lista_cursos)
          CASO 3:
            IMPRIMIR "Promedio general:", promedio_general(lista_cursos)
          CASO 4:
            aprobado, reprobado = cursos_aprobados(lista_cursos)
            IMPRIMIR "Cursos aprobados:", aprobado, "Cursos reprobados:", reprobado
          CASO 5:
            SI lista_curso = vacía ENTONCES
                    IMPRIMIR “No hay cursos registrados”
            LEER busqueda
            mostrar_busqueda(buscar_curso(lista_cursos, busqueda), busqueda)
          CASO 6:
            LEER busqueda
            encontrado = buscar_curso(lista_cursos, busqueda)
            SI encontrado ≠ [] ENTONCES
              LEER nueva_nota
              actualizar_nota(encontrado, nueva_nota)
```

```
            FIN SI
        CASO 7:
            LEER busqueda
            eliminar_curso(lista_cursos, busqueda)
        CASO 8:
            ordenamiento_por_nota(lista_cursos)
        CASO 9:
            ordenamiento_por_nombre(lista_cursos)
        CASO 10:
            LEER busqueda
            busqueda_por_nombre_binario(lista_cursos, busqueda)
        CASO 11:
            solicitud_de_revision()
        CASO 12:
            mostrar_historial()
        CASO 13:
            IMPRIMIR "Fin del programa"
            SALIR DEL BUCLE
    FIN SEGÚN
  FIN MIENTRAS

FIN
```