UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS
CIENCIAS DE LA COMPUTACIÓN 1
PROYECTO FINAL 2024-1

Enunciado del problema.

El director del área Ciencias de la Computación, de Ingeniería de sistemas desea realizar un seguimiento a los cursos que maneja en su área. Para ello, le ha solicitado a cada profesor que le envíe la información de las notas de cada uno de los parciales del semestre, relacionando la calificación obtenida en cada uno de los puntos por cada estudiante.

ESTRUCTURA DE DATOS

A continuación, se describen los archivos y estructuras que requiere el problema para ser solucionado:

Siguiendo las indicaciones del director, cada profesor envía durante el semestre tres archivos, cada uno de los cuales tienen en la primera línea el número del parcial y las siguientes líneas contienen las notas obtenidas por cada estudiante. Lo anterior significa que hay tantas líneas como estudiantes haya en el curso, y en cada línea hay tantas notas como puntos haya en el parcial. Un ejemplo del archivo es:

Nombre del archivo: CCI82.doc (corresponde al primer parcial del grupo 82 de ciencias de la computación I, el cual tiene tres estudiantes y cuyo parcial tuvo tres puntos, cada uno con notas que oscilan entre 0.0 y 5.0)

```
1
María Rodríguez   4.0   2.5   4.0
Juan Pérez        3.2   2.3   4.6
José González           1.5   5.0   4.2
```

Por su parte, para manejar fácilmente la información, el director de área necesita crear:

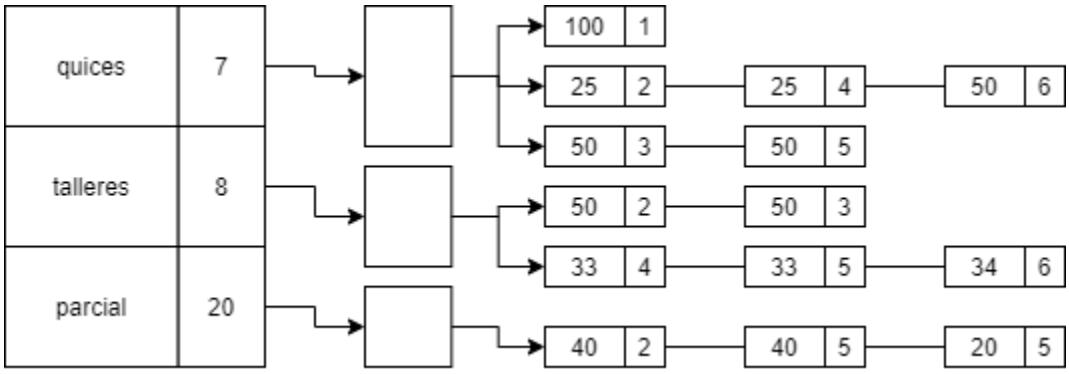
1. Una lista dinámica donde el apuntador al primer elemento de la lista, se llamará "**Profesores**". Cada uno de los elementos de la lista almacena la cédula, los apellidos, los nombres y la cantidad de clases que tiene asignadas cada profesor. Un ejemplo de cada elemento de la lista sería.

cédula	Apellidos del Profesor	Nombres del Profesor	Número de clases
19234568	Gómez Parra	Juan Pablo	2

cédula	Apellidos del Profesor	Nombres del Profesor	Número de clases
79458124	Perea Duarte	Carlos	1

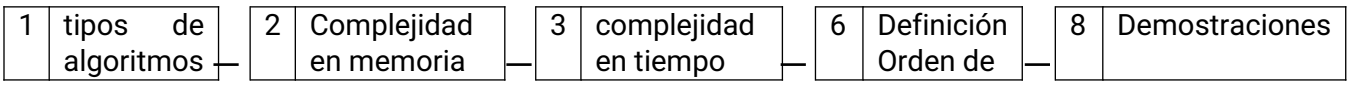
2. Cada profesor tendrá asociada una lista de arreglos llamada **cortes de notas** donde cada

elemento de la lista apunta a un arreglo que contiene las notas del corte correspondiente, es decir, si hay tres cortes, la lista tendrá 3 elementos cada uno apuntando a un arreglo diferente. Cada una de las posiciones del arreglo tiene tres campos, el primero de ellos corresponde al tipo de evaluación (quiz, taller, parcial, proyecto, exposición etc), el segundo al porcentaje total de dicha evaluación y el tercero tiene asociada una o varias listas de parejas. El primer valor de la pareja indica el porcentaje de cada punto del parcial, quiz o taller y el segundo valor indica el código del tema evaluado. Es decir, el número de elementos del arreglo es igual al número de tipos de evaluación para cada corte y por cada punto en el parcial o en el quiz habrá un elemento en la lista. Ejemplo

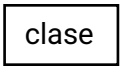


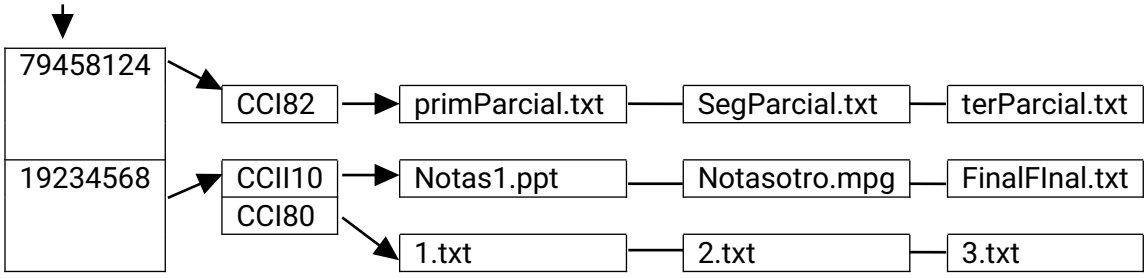
En el ejemplo se puede observar que en el corte hubo tres tipos de evaluación: quices (7%), talleres (8%) y parcial (20%). Se evidencia también que hubo tres quices, dos talleres y un parcial El primer quiz tuvo solamente un punto con 100% que evaluaba el tema 1, el segundo quiz un punto con 25% que evaluaba el tema 2, otro punto con el 25% que evaluaba el tema 4 y un punto con el 50% que evaluaba el punto 6. Y el tercer quiz tuvo dos puntos cada uno con el 50% el primer punto evaluaba el tema 3 y el segundo el tema 5. Por otra parte, hubo dos talleres el primero evaluaba dos temas (el 2 y el 3) cada uno con valor del 50% el segundo taller evaluaba tres temas (4,5 y 6) los dos primeros temas con el 33% cada uno y el último con el 34% Finalmente se realizó un parcial el cual tuvo 3 puntos, el primero y el segundo que evaluaban los temas 2 y 5, con un valor del 40% cada uno y el tercero con valor del 20% evaluando el tema 5.

3. Una lista llamada **temas,** en la cual cada elemento está compuesto por dos campos: el código del tema y su respectivo nombre. Ejemplo de la lista:



4. Un arreglo llamado **Clases**, donde el número de elementos del arreglo se corresponde con el número de profesores de la asignatura (cada posición tiene la cédula del profesor y un apuntador al arreglo). A su vez, cada profesor tiene asociado arreglo de tamaño igual al número de cursos que tiene a su cargo (dato que se puede consultar en la lista profesores), en cada posición de dicho arreglo se tiene el código de la clase dictada por el profesor y un apuntador a una lista ordenada con el nombre de los archivos que envió el profesor(el primer nombre corresponde al primer parcial, el segundo al segundo y así sucesivamente) con el consolidado de sus notas (El número de elementos de la lista corresponde a la cantidad de parciales que hubo en el semestre) Ejemplo:





En el ejemplo se observa que el primer profesor tiene asignada la clase CCI82 y el segundo profesor tiene asignadas las clases CCII10 y CCI80. Cada uno de ellos envió para cada una de sus clases, tres archivos con las notas de cada uno de los parciales. Estos archivos cumplen con la descripción dada en la primera parte de este documento.

Estructuras en memoria secundaria:

Aunque por petición del director se crean una serie de estructuras de datos, la información deberá estar almacenada, de manera permanente, en archivos planos los cuales deben ser actualizados al final de la ejecución, dado que, por ejemplo, durante el proceso puede ser excluido o incluido: un profesor, un estudiante, un tema, una evaluación o un punto de la evaluación. Debe tenerse la posibilidad de modificar la información ya sea agregando, eliminando o cambiando registros.

Existen por lo menos los archivos planos de parciales, profesores, cortes de notas, temas y clases.

Estructuras en memoria Principal:

Además de las listas ya planteadas y con el fin de optimizar las búsquedas, deberán utilizarse listas, multilistas, arreglos, pilas, colas o árboles en memoria principal. Para la realización de dichas búsquedas, deberá evitarse la utilización total de la información. Es decir, se espera que apliquen los criterios y conocimientos correspondientes a la eficiencia de los algoritmos. Complejidad en tiempo y memoria.

REQUERIMIENTOS FUNCIONALES.

Elabore un programa donde:

1. Se creen las estructuras aquí mencionadas, que el director del área requiere.
2. Se desarrollen herramientas para que el Jefe de sección pueda llenar estas estructuras
3. Se almacenen las estructuras en archivos, de forma que cuando se termine de trabajar con ellas, la información allí contenida no se pierda y que la próxima ocasión que se utilice el programa, las estructuras aparezcan ya llenas con la información que recuperan de dichos archivos.
4. Se generen archivos donde se consolida la información para cada uno de los parciales hechos en el semestre. El nombre de los archivos generados debe ser ParcialXX.bin, donde xx debe ser reemplazado por el número del parcial. En cada archivo deberá ir el número de la pregunta y la nota promedio de todos los estudiantes inscritos.
5. Se cree un archivo llamado **refuerzo.txt,** ordenado ascendentemente por el campo nota donde se puede visualizar en qué temas hay que hacer refuerzo para el próximo semestre (consolidado de todos los cursos durante todo el semestre para todos los parciales). Ejemplo:

Funciones	2.3
Vectores	3.2

6. Se permita al director de área, consultar:
    a. El promedio de un estudiante perteneciente a un curso específico identificando además el comportamiento en los diferentes temas por parte de dicho estudiante.
    b. El comportamiento promedio de los estudiantes pertenecientes a un curso dado, en los diferentes temas.
    c. El comportamiento promedio (clasificado por temas) de todos los estudiantes que toman clase con el mismo profesor.
    d. La lista (ordenada por cédula) de los profesores que tienen un número de clases dado (por consola).
    e. Lista de los estudiantes (pertenecientes a cualquiera de los cursos) que tienen nota superior a una nota dada, en las preguntas de un tema determinado por el usuario.
7. Se permitirá al profesor modificar el esquema definido para cada uno de los cortes de notas, siempre y cuando la evaluación que quiera modificar, no se halla llevado a cabo aún.
8. Las funciones deben poderse utilizar sin importar el número de profesores, parciales. quices, talleres, puntos, temas, etc. Es decir, no se debe crear el programa solo para los ejemplos dados.
9. EL programa contará con los menús y submenús que faciliten la realización de cada una de las consultas.
10. Se debe aplicar programación orientada a objetos y el principio de alta cohesión y bajo acoplamiento.
11. Se deben aplicar los conceptos de optimización estudiados en el curso. No limitarse a garantizar el funcionamiento del programa, sin embargo, se podrá usar información redundante (repetida) sólo en los casos que sea estrictamente necesario y que tengan soporte argumentativo que demuestre una mejora sustancial, en términos de comportamiento de los algoritmos.

Condiciones de entrega

1. Se puede elaborar en grupos de por lo menos dos y máximo tres personas (Es un trabajo en grupo). Se sustentará individualmente.
2. Se entrega a través de aula virtual una carpeta comprimida con: código fuente, ejecutable, archivos planos y documento soporte. Sencillo manual para ejecución (si se requiere).
3. El programa debe correr en DevC++, sin requerir **ningún tipo de ajuste.**
4. Documento soporte incluye: Diseño de la solución, Contenido y estructura de archivos planos, Estructuras de datos en memoria principal definidas, incluyendo diagrama.
5. Archivo fuente debidamente autodocumentado.
6. Los archivos planos contendrán por lo menos: 10 cursos (cada profesor tendrá asignado entre 1 y 3 clases), 5 estudiantes por curso y 10 notas por estudiante.

Docente: *Deicy Alvarado.*