Estructura de datos y de la información
Boletín de problemas - Tema 8

1. Implementar una clase lista utilizando un vector. Los elementos de la lista se guardan en las posiciones inferiores del vector en el mismo orden que ocupan en la lista. De este modo, para insertar un nuevo elemento en medio de la lista será necesario *abrir el hueco* correspondiente, y al borrar un elemento habrá que desplazar los restantes para *cerrar el hueco* generado. Calcular el coste de las distintas operaciones.

2. Añadir a la clase lista dinámica con simple enlace una operación que inserte un elemento después de una posición dada. Utilizar la siguiente cabecera:
   `apuntador insert_after(apuntador pos, const T &valor)` .

3. Modificar la clase lista con simple enlace para disponer en todo momento de un puntero al primer elemento y de otro al último. ¿Se modifica el orden del coste de alguna operación?

4. Implementar una clase lista circular con simple enlace. Para ello habrá que modificar la clase lista dinámica con simple enlace para que el último nodo se encuentre siempre enlazado con el primero.

5. Implementar una clase lista con simple enlace. La lista contendrá siempre al menos un nodo vacío que denominaremos *nodo cabecera*. ¿Pueden simplificarse las operaciones de inserción y borrado?

6. Implementar una clase `deque` utilizando una estructura dinámica doblemente enlazada que incorpore un puntero al frente y otro al final. Comparar el coste de las distintas operaciones con las de la implementación simplemente enlazada del tema anterior.

7. Dada una clase lista dinámica con simple enlace, implementar una operación que invierta la posición de los elementos de una lista.

   a) Suponer que la operación forma parte de la clase y modificar tan sólo los punteros de la lista.

   b) Suponer que la operación es externa a la clase y utilizar tan sólo operaciones de la misma.

8. Supongamos que la clase lista no incorpora la operación que busca un elemento dado y devuelve su posición. Implementar dicha operación fuera de la clase y utilizando sus operaciones. El prototipo de dicha operación será:
   `apuntador Buscar(const lista<T> &l, const  T &valor);`

9. Implementar la clase pila usando las operaciones de la clase lista.

10. Implementar la clase cola usando las operaciones de la clase lista.

11. Se tiene implementada una lista de empleados de una empresa ordenada alfabéticamente por sus apellidos. Se desea reducir el tiempo de búsqueda de los datos de un empleado y para ello, el programador decide ampliar la estructura de datos añadiendo un vector cuyo índice son las iniciales de los apellidos, y cuyos elementos son posiciones en la lista del primer empleado cuyo apellido empiece por dicha inicial. Si para alguna letra no hay ningún empleado, la posición guardada será `NULL`. Se pide:

    a) La definición de tipos necesaria.

    b) Determinar qué operaciones de lista ordenada se pueden modificar para mejorar su eficiencia con esta nueva estructura y cuáles no.

    c) Escribir las nuevas operaciones de lista (sólo las mejorables).

12. Se dispone de una lista de simple enlace en la que la información almacenada es de tipo *string*. La lista está ordenada por orden alfabético.
    Se pretende utilizar la lista para gestionar el orden en el que los distintos candidatos han de realizar una oposición. El mecanismo habitual es seguir el orden alfabético, pero no se comienza por la letra A, sino por una letra extraída al azar: si, por ejemplo, sale la L, el ejercicio lo comienza el primer candidato cuyo apellido empiece por L, siguiéndose, a partir de ahí, el orden alfabético. Al acabar la lista, se continúa con el primero hasta que todos los candidatos se hayan examinado.
    Sabiendo que nos darán como información de entrada la letra que salga por sorteo, se pide:

    a) Implementar un algoritmo que a partir de la lista original, ordenada, y de la letra, devuelva *otra lista* con la reordenación indicada.

    b) Implementar un algoritmo que a partir de la lista original, ordenada, y de la letra, devuelva *la misma lista* pero con la reordenación indicada.

13. Dada una lista doblemente enlazada, que almacena información de tipo base `tb`, definir la operación `borra_2`, tal que dada la posición de un elemento borre el anterior y el siguiente (si un elemento dado no tuviera anterior y/o siguiente, que borre lo que pueda).

14. Dada una lista, desordenada, que almacena información de tipo entero, escribir un algoritmo que obtenga dos listas ordenadas: en una aparecerán los elementos pares de la lista original y en la otra los impares.

15. Una finca tiene N pisos y en cada piso hay 5 viviendas identificadas como A, B, C, D y E. Por cada vivienda, un vecino paga una determinada cuota de gastos de escalera. El administrador de la finca sabe el nombre de cada vecino, su cuota y, además, si está alquilado o si es el propietario del piso; en el primer caso, debe saber, además, el nombre del propietario del piso y su número de cuenta corriente, puesto que es el propietario el que corre con los gastos de escalera. En el segundo caso, para poder realizar el cargo debe saber el año en que se compró el piso y el número de cuenta del propietario.

    a) Definir la estructura de datos que permita organizar dicha información.

b) Escribir una operación que permita al administrador sacar un listado en el que, para cada una de las viviendas, se emita un recibo. Un recibo indica el piso, la puerta, el nombre del vecino, el nombre del propietario, la cuota y el número de cuenta donde se deben efectuar los cargos del recibo de la escalera.

c) Al cabo de 15 días desde la emisión del listado y su envío al banco con el que trabaja el administrador, el banco devuelve una lista de recibos impagados, que el administrador transforma en una lista de morosos (teniendo en cuenta que puede haber alguien que posea más de una vivienda en la finca).

1) Definir las estructuras para guardar lista de recibos y lista de morosos. En la lista de morosos, debe constar el nombre, la cuenta corriente y el importe total de la deuda.
2) Escribir una operación que permita transformar la lista de recibos impagados en la lista de morosos, sin modificar la lista de recibos.

16. En un teatro se identifica cada butaca por su número de fila y su número de butaca dentro de una fila. En total hay N filas y M butacas por fila. Se quiere hacer un programa para el proceso automático de reservas del teatro, de forma que por cada butaca, además de su precio, si está reservada se pueda saber quién la ha reservado.

a) Definir la estructura de datos `Tteatro` para guardar la información sobre las butacas.

b) Para realizar una reserva, además del nombre de quien la hace, hay que indicar cuántos asientos se reservan. Las reservas se deben almacenar en una lista, implementada dinámicamente que servirá, además, para cobrar la reserva en el momento del pago.

1) Definir la estructura de datos `Treserva` para guardar las reservas.
2) Escribir la operación `Reservar` que, dada la información del nombre y número de asientos, actualice la ocupación del teatro y además añada dicha información a la lista de reservas, incluyendo el precio. La reserva debe asegurar que los asientos sean correlativos. Si no es posible, debe indicarlo.
3) Escribir la operación `Cancelar` que, dada la información del nombre, permite cancelar una reserva, eliminándola de la lista y liberando, además, los asientos correspondientes.

17. En una determinada Universidad hay N campus, cada uno de ellos con M aulas informáticas y cada aula con 20 puestos de trabajo. Cada uno de estos puestos de trabajo puede estar ocupado o no con una conexión a la red. Caso de estarlo, el usuario quedará identificado por la siguiente información: su *login* y la tarea que está realizando. La tarea puede ser una sola de estas cuatro: leer las *news*, hacer un *write*, acceder al correo electrónico o manejo del Sistema Operativo.
Se desea escribir un algoritmo que permita a los operadores del sistema saber quién está conectado a la red; para ello, se quiere

a) La definición de la estructura de datos que soporte la información descrita.

b) Definir la estructura de datos más adecuada para contener la salida de datos, si lo que se quiere es la relación de los *logins* de usuarios conectados. Escribir el algoritmo que permita obtener dicha relación.

c) Definir la estructura de datos más adecuada para contener la salida de datos, si lo que se quiere es la relación de los *logins*, tarea, ordenador, aula informática y campus de los usuarios conectados. Escribir el algoritmo que permita obtener dicha relación.

Nota: Un *login* es una identificación única. Se supone que como mucho hay un número máximo de L *logins*, L < 256.

18. Un profesor tiene un montón de exámenes y en cada momento sólo puede acceder al examen situado encima del mismo. Por cada uno de ellos guarda el nombre del alumno, su dni y la nota de cada uno de los cinco ejercicios de que consta el examen.

    a) Definir las estructuras de datos más adecuadas para almacenar la información del problema.

    b) El profesor quiere generar el acta en la que los distintos alumnos aparecen ordenados alfabéticamente. Por cada alumno debe guardar entonces su nombre, dni y la nota total obtenida sumando la de los 5 ejercicios.

        1) Definir las estructuras de datos más adecuadas para guardar la información.
        2) Escribir un algoritmo `publicar` que a partir del montón de exámenes genere el acta. Al finalizar el proceso, el montón de exámenes debe quedar igual.

    c) Escribir un algoritmo denominado `maxnota` que, a partir del acta, escriba el nombre y la nota del alumno que haya obtenido la máxima calificación global.

    d) Suponer que el profesor quiere poder recorrer el acta, tanto por orden alfabético como por orden de la nota global de los distintos alumnos.

        1) Modificar la estructura de datos del segundo apartado para poder realizar esta operación.
        2) Escribir el algoritmo `recorrer` que, en función de un argumento de entrada, permita recorrer el acta por orden de nombre o de nota. Por cada alumno, el algoritmo debe escribir su nombre y la nota total.

19. En una sucursal bancaria se pueden realizar ingresos, reintegros y consultas. La sucursal dispone de dos ventanillas, una de las cuales puede atender cualquier operación (ventanilla general), mientras que la otra sólo puede realizar ingresos (ventanilla de ingresos). Los clientes, caracterizados por su NIF y la operación que desean realizar, se atienden según el orden de llegada, y pueden ser atendidos indistintamente por una u otra ventanilla en función de que estén libres en ese momento.
    Si un cliente es atendido por la ventanilla de ingresos y no desea realizar esa

operación, se pasa a una dependencia donde esperan en orden de llegada todos los clientes en la misma situación. Todos los clientes de esta dependencia serán atendidos por la ventanilla general. Además, tienen preferencia sobre los clientes que aún no han visitado ninguna ventanilla.
Cada operación realizada en la sucursal se registra ordenada por NIF del cliente en una estructura denominada `Operaciones` en el momento de efectuarla. Por cada operación se almacena el NIF del cliente y la operación realizada.

a) Definir las estructuras de datos más adecuadas para almacenar la información del problema.

b) Escribir las operaciones para atender un cliente en la ventanilla general y atenderlo en la ventanilla de ingresos.

c) Al finalizar la jornada se reordenan las operaciones efectuadas colocando en primer lugar los ingresos, luego los reintegros y en tercer lugar las consultas, y dentro de cada modalidad manteniendo el orden por NIF del cliente. Se pide realizar un algoritmo que a partir de la estructura `Operaciones` obtenga la estructura `Operaciones_por_modalidad`.

20. Escribir un algoritmo que permita intercambiar los elementos que ocupan las posiciones m y n de una lista dinámica de elementos de `tipobase`. Tener en cuenta que no puede modificarse el valor contenido en los nodos de la lista y que $1 \leq m \leq n \leq final$ de la lista.