Introducción a las Bases de Datos

Fundamentos de Organización de Datos

Práctica 3

Archivos Secuenciales - Bajas

1. Modificar el ejercicio 4 de la práctica 1 (programa de gestión de empleados), agregándole una opción para realizar bajas copiando el último registro del archivo en la posición del registro a borrar y luego truncando el archivo en la posición del último registro de forma tal de evitar duplicados.

2. Definir un programa que genere un archivo con registros de longitud fija conteniendo información de asistentes a un congreso a partir de la información obtenida por teclado. Se deberá almacenar la siguiente información: nro de asistente, apellido y nombre, email, teléfono y D.N.I. Implementar un procedimiento que, a partir del archivo de datos generado, elimine de forma lógica todos los asistentes con nro de asistente inferior a 1000.

   Para ello se podrá utilizar algún carácter especial situándolo delante de algún campo String a su elección. Ejemplo: '@Saldaño'.

3. Realizar un programa que genere un archivo de novelas filmadas durante el presente año. De cada novela se registra: código, género, nombre, duración, director y precio. El programa debe presentar un menú con las siguientes opciones:

a. Crear el archivo y cargarlo a partir de datos ingresados por teclado. Se utiliza la técnica de lista invertida para recuperar espacio libre en el archivo. Para ello, durante la creación del archivo, en el primer registro del mismo se debe almacenar la cabecera de la lista. Es decir un registro ficticio, inicializando con el valor cero (0) el campo correspondiente al código de novela, el cual indica que no hay espacio libre dentro del archivo.

b. Abrir el archivo existente y permitir su mantenimiento teniendo en cuenta el inciso a., se utiliza lista invertida para recuperación de espacio. En particular, para el campo de ´enlace´ de la lista, se debe especificar los números de registro referenciados con signo negativo, (utilice el código de novela como enlace).Una vez abierto el archivo, brindar operaciones para:

    i. Dar de alta una novela leyendo la información desde teclado. Para esta operación, en caso de ser posible, deberá recuperarse el espacio libre. Es decir, si en el campo correspondiente al código de novela del registro cabecera hay un valor negativo, por ejemplo -5, se debe leer el registro en la posición 5, copiarlo en la posición 0 (actualizar la lista de espacio libre) y grabar el nuevo registro en la posición 5. Con el valor 0 (cero) en el registro cabecera se indica que no hay espacio libre.

    ii. Modificar los datos de una novela leyendo la información desde teclado. El código de novela no puede ser modificado.

    iii. Eliminar una novela cuyo código es ingresado por teclado. Por ejemplo, si se da de baja un registro en la posición 8, en el campo código de novela del registro cabecera deberá figurar -8, y en el registro en la posición 8 debe copiarse el antiguo registro cabecera.

c. Listar en un archivo de texto todas las novelas, incluyendo las borradas, que representan la lista de espacio libre. El archivo debe llamarse "*novelas.txt*".

NOTA: Tanto en la creación como en la apertura el nombre del archivo debe ser proporcionado por el usuario.

4. Dada la siguiente estructura:

```
type


     reg_flor = record
                  nombre: String[45];
                  codigo:integer;

     tArchFlores = file of reg_flor;
```

Las bajas se realizan apilando registros borrados y las altas reutilizando registros borrados. El registro 0 se usa como cabecera de la pila de registros borrados: el número 0 en el campo código implica que no hay registros borrados y -N indica que el próximo registro a reutilizar es el N, siendo éste un número relativo de registro válido.

a. Implemente el siguiente módulo:

```
{Abre el archivo y agrega una flor, recibida como parámetro manteniendo la política descripta anteriormente}

procedure agregarFlor (var a: tArchFlores ; nombre: string; codigo:integer);
```

b. Liste el contenido del archivo omitiendo las flores eliminadas. Modifique lo que considere necesario para obtener el listado.

5. Dada la estructura planteada en el ejercicio anterior, implemente el siguiente módulo:

```
{Abre el archivo y elimina la flor recibida como parámetro manteniendo la política descripta anteriormente}

procedure eliminarFlor (var a: tArchFlores; flor:reg_flor);
```

6. Una cadena de tiendas de indumentaria posee un archivo maestro no ordenado con la información correspondiente a las prendas que se encuentran a la venta. De cada prenda se registra: cod_prenda, descripción, colores, tipo_prenda, stock y precio_unitario. Ante un eventual cambio de temporada, se deben actualizar las prendas a la venta. Para ello reciben un archivo conteniendo: cod_prenda de las prendas que

quedarán obsoletas. Deberá implementar un procedimiento que reciba ambos archivos y realice la baja lógica de las prendas, para ello deberá modificar el stock de la prenda correspondiente a valor negativo.
Por último, una vez finalizadas las bajas lógicas, deberá efectivizar las mismas compactando el archivo. Para ello se deberá utilizar una estructura auxiliar, renombrando el archivo original al finalizar el proceso.. Solo deben quedar en el archivo las prendas que no fueron borradas, una vez realizadas todas las bajas físicas.

7. Se cuenta con un archivo que almacena información sobre especies de aves en vía de extinción, para ello se almacena: código, nombre de la especie, familia de ave, descripción y zona geográfica. El archivo no está ordenado por ningún criterio. Realice un programa que elimine especies de aves, para ello se recibe por teclado las especies a eliminar. Deberá realizar todas las declaraciones necesarias, implementar todos los procedimientos que requiera y una alternativa para borrar los registros. Para ello deberá implementar dos procedimientos, uno que marque los registros a borrar y posteriormente otro procedimiento que compacte el archivo, quitando los registros marcados. Para quitar los registros se deberá copiar el último registro del archivo en la posición del registro a borrar y luego eliminar del archivo el último registro de forma tal de evitar registros duplicados.

Nota: Las bajas deben finalizar al recibir el código 500000

8. Se cuenta con un archivo con información de las diferentes distribuciones de linux existentes. De cada distribución se conoce: nombre, año de lanzamiento, número de versión del kernel, cantidad de desarrolladores y descripción. El nombre de las distribuciones no puede repetirse.
Este archivo debe ser mantenido realizando bajas lógicas y utilizando la técnica de reutilización de espacio libre llamada **lista invertida**.
Escriba la definición de las estructuras de datos necesarias y los siguientes procedimientos:
ExisteDistribucion: módulo que recibe por parámetro un nombre y devuelve verdadero si la distribución existe en el archivo o falso en caso contrario.
AltaDistribución: módulo que lee por teclado los datos de una nueva distribución y la agrega al archivo reutilizando espacio disponible en caso de que exista. (El control de

unicidad lo debe realizar utilizando el módulo anterior). En caso de que la distribución que se quiere agregar ya exista se debe informar "ya existe la distribución".

BajaDistribución: módulo que da de baja lógicamente una distribución cuyo nombre se lee por teclado. Para marcar una distribución como borrada se debe utilizar el campo cantidad de desarrolladores para mantener actualizada la lista invertida. Para verificar que la distribución a borrar exista debe utilizar el módulo ExisteDistribucion. En caso de no existir se debe informar "Distribución no existente".