ALGORITMOS Y ESTRUCTURAS DE DATOS

TRABAJO PRÁCTICO

Algoritmos y Estructuras de Datos F.C es un equipo de fútbol fundado en 1988, y desde su nacimiento participa de las ligas menores de Argentina. Bajo la coordinación de su último presidente, el club ascendió hasta la segunda división y con ello su popularidad en todo el país, lo cual genera que todos los meses sus socios sigan aumentando.

Ante este crecimiento, el sector Administrativo del club se encuentra en la búsqueda de ***un sistema que les permita encargarse de manera más organizada la gestión de los socios***. Actualmente la información de los mismos es manejada desde Excel con el siguiente formato:

Campo	Tipo	
DNI (Clave)	Long	10.000 < DNI < 100.000.000
Apellidos	Cadena (60)	
Nombres	Cadena (60)	
Fecha de Nacimiento	T_Fecha	
Sexo	Char	'F' (Femenino), 'M' (Masculino) u 'O' (Otro)
Fecha de afiliación	T_Fecha	
Categoría	Cadena (10)	'MENOR', 'CADETE', 'ADULTO', 'VITALICIO', 'HONORARIO', 'JUBILADO'. Validar [(1)]
Fecha de última cuota paga	T_Fecha	
Estado	Char	'A' Activo, 'I' Inactivo - Se genera en el alta con 'A'
Fecha de Baja	T_Fecha	Si el estado es 'A', éste es vacío

Nuestro programa va a recibir el Excel exportado a csv con registros de longitud variable y los campos separados por "," y NO se encuentra ordenado por ningún campo en particular.

Biblioteca de Índice

Construir un Tipo de Dato Abstracto (TDA) Índice (t_indice) generico, donde en cada elemento o entrada de este se almacena una clave que puede ser de cualquier tipo (void *clave) y el número de registro (unsigned nro_reg) que le corresponde a esa entrada de índice en el archivo de datos original.

Este TDA índice podría ser implementado sobre diferentes estructuras de datos, como en un array estático, un array redimensionable, una lista simplemente enlazada, una lista doblemente enlazada, un árbol o algunas otras más. Para este trabajo práctico se pide hacer la implementación utilizando un árbol. Hay 2 formas de implementarlo usando la estructura de datos árbol: la primera sería utilizando el TDA Árbol y todas sus primitivas, llamándolas desde las primitivas de índice, y la segunda sería implementando lo necesario para el manejo de árboles dentro de las primitivas del índice. Puede hacerlo de cualquiera de los métodos antes mencionados.

Las primitivas del TDA son:

- **void ind_crear (t_indice* ind, size_t tam_clave, int (*cmp)(const void*, const void*))**: inicializa la estructura a índice vacío y almacena en la estructura de índice el tamaño de la clave genérica a utilizar y la función de comparación.
- **int ind_insertar (t_indice* ind, void *clave, unsigned nro_reg)**: inserta en orden el registro reg_ind según la clave. [(1)]
- **int ind_eliminar (t_indice* ind, void *clave, unsigned *nro_reg)**: elimina la entrada del índice correspondiente a la clave y devuelve en nro_reg el número de registro asociado. [(1)]
- **int ind_buscar (const t_indice* ind, void *clave, unsigned *nro_reg)**: Busca la clave recibida por parámetro y devuelve en nro_reg el número de registro asociado. [(1)]
- **int ind_cargar (t_indice* ind, const char* path):** Carga el índice a partir de un archivo binario ordenado (típicamente de extensión '.idx'), donde cada registro del archivo tiene la estructura definida, clave-nro_reg. [(1)]
- **int ind_grabar (const t_indice* ind, const char* path)**: Graba un archivo binario ordenado (típicamente de extensión '.idx') con el contenido del índice con la estructura clave-nro_reg. [(1)]
- **void ind_vaciar (t_indice* ind)**: deja el índice en su estado de vacío.
- **int ind_recorrer (const t_indice* ind, void (*accion)(const void *, unsigned, void *), void* param)**: Recorre el índice en orden y llama a acción para cada registro del mismo. [(1)]

[(1)] : Devuelve 1 (uno) si la operación fue exitosa y 0 (cero) en caso contrario.

Primer aplicación: Generar el índice

Construir un programa que lea el archivo de texto "socios.csv" y genere el archivo "socios.dat" transformando los datos a binarios. Para ello, el usuario debe ingresar por teclado el path (ruta de acceso) en el que se encuentra el archivo.

Una vez finalizada la creación del archivo"socios.dat" construir un índice donde la clave sea el DNI del Socio, y el el número de registro (unsigned nro_reg) que le corresponde a ese socio en el archivo.

Antes de finalizar el programa, guardar el índice generado en el archivo "socios.idx".

Segunda aplicación: Gestión de Socios

Una vez depurada la información histórica de socios y generado el índice, mostrar por pantalla un menú que permita al usuario:

- ✔ (A) Dar de alta un nuevo socio solicitando todos los campos de la estructura socios. Verificar previamente que el DNI ingresado no exista anteriormente en el club con estado distinto a 'B'. Cabe destacar que cualquier registro con estado 'B' para el programa no existe, y si se da de alta un nuevo Socio que ya existía, pero como estado 'B', debe ser dado de alta como un nuevo socio. Insertar al nuevo socio en el índice.
- ✔ (M) Modificar la información de un socio, "Apellido", "Nombre", "Categoria", "Sexo" o "fecha de ultima cuota paga" a partir del DNI del socio a buscar. Validar el formato del campo según la estructura inicial.
- ✔ (B) Dar de baja un socio por DNI. Si existe, cambiar el estado a 'B' y guardar en el campo Fecha de Baja la fecha de hoy. Además debe eliminar el socio del índice. Si no existe o ya está dado de baja, también informar por pantalla.
- ✔ (L) Listar todos los socios ordenados (sin mostrar los dados de baja) por DNI.
- ✔ (P) Listar los 10 socios con mayor retraso en la fecha de pago de la cuota.
- ✔ (S) Salir

Una vez que el usuario seleccione Salir, No olvidar:

- Grabar el archivo de índices.
- Liberar la memoria del índice.