**Tópicos de Programación TRABAJO PRÁCTICO**

**El club de futbol barrial “La tradición”** desde su nacimiento participa de las ligas interclubes de Argentina. Con mucha dedicación y práctica de todo el club, durante los últimos años ascendió a la segunda división y con ello su popularidad en todo el país, lo que genera que todos los meses sus socios sean cada vez más.

Ante este crecimiento, el sector Administrativo del club solicitó la creación de ***una aplicación que les permita encargarse de manera más organizada la gestión de los socios***.

# **Primera parte**

Construir un programa que genere un archivo binario (socios.dat) a partir de uno de texto (socios.txt) que tiene registros de longitud variable, validando la información de acuerdo a lo detallado más abajo. Esta validación debe hacerse una vez convertido todo un registro y, si se detecta algún error, grabar en otro archivo de texto de salida la línea de texto completa e ignorar el registro.

Al comenzar, debe ingresarse la fecha de proceso y el path del archivo (mi\_ruta/socios.txt).

Para ello se define la estructura de información que se desea tratar:

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** |  |
| DNI (Clave) | long | 10.000 < DNI < 100.000.000 |
| Apellidos y Nombres | char (60) |  |
| Fecha de Nacimiento | t\_fecha | Validación formal y < fecha de proceso – 10 años |
| Sexo | char | ‘F’ (Femenino), ‘M’ (Masculino) u ‘O’ (Otro) |
| Fecha de afiliación | t\_fecha | Validación formal, <= fecha de proceso y > fecha  nacimiento |
| Categoría | char (10) | ‘MENOR’, ‘CADETE’, ‘ADULTO’, ‘VITALICIO’,  ‘HONORARIO’, ‘JUBILADO’. Validar |
| Fecha de última cuota paga | t\_fecha | > fecha de afiliación y <= fecha de proceso. Si no se  informa, asumir la fecha de afiliación. |
| 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 |

Para el caso de los apellidos y nombres, deberá realizarse una normalización de estos, consistente en:

* El/Los apellido/s y nombre/s deben comenzar con letra mayúscula y luego continuar con minúscula.
* El/Los apellido/s deben estar separado/s del/los nombre/s por una coma. De no existir dicha coma, agregarla a continuación de la primera palabra.
* Cada palabra debe separarse por no más de un espacio.

ejemplo:

galletA pePE raul  Galleta, Pepe Raul

# **Segunda parte**

Crear un menú que permita actualizar el archivo binario socios.dat, generado en el punto anterior que contenga las siguientes opciones:

1. Alta
2. Baja
3. Modificación
4. Mostrar información de un socio
5. Listado de socios ordenados por clave (DNI)
6. Salir

Para esto deberá valerse del uso de un TDA Índice implementado en un array estático o con asignación dinámica de memoria.

**Alta:** se obtendrán los datos del teclado, ingresando primero la clave (DNI) verificando que no exista en el índice y, cuando estén todos los datos ingresados, se realizará la validación y consistencia de estos (ídem proceso de generación del archivo). Si se detectan errores, se ignora todo lo ingresado. Una vez aceptado, grabarlo al final del archivo binario e insertar la información correspondiente en el índice.

**Baja:** Si existe la clave que se quiere dar de baja y estado no es ‘I’, actualizar el registro correspondiente con un carácter ‘I’ en estado y con la fecha de proceso como fecha de baja. Eliminar registro correspondiente del índice.

**Modificación**: Si existe la clave que se quiere modificar y el estado del registro no es ‘I’, actualizar la

información deseada. Si se detectan errores, se ignora todo lo ingresado.

**Mostrar información de un socio:** Si existe la clave que se quiere dar de baja y estado no es ‘I’, mostrar toda la información correspondiente al socio.

**Listado de socios ordenados por clave (DNI):** Debe mostrar todos los registros activos (no están dados de baja) ordenados por dni.

**Salir:** Debe terminar la ejecución del programa

Como procedimientos finales, debe liberar la memoria del índice.

# **TDA Indice (Array Estático)**

Construir un TDA indice (t\_indice) en los archivos indice.h e indice.c, cuya estructura es un array donde cada elemento tiene la estructura {long dni; unsigned nro\_reg;} (t\_reg\_indice). Cada entrada del índice (dni-nro\_reg) se guarda en cada una de las posiciones del array.

Las primitivas del TDA son:

void indice\_crear ( t\_indice\*): inicializa la estructura a índice vacio.

int indice\_insertar (t\_indice\*, const t\_reg\_indice\*): inserta en orden según el dni. int indice\_eliminar (t\_indice\*, t\_reg\_indice\*): elimina el dni informado.

int indice\_vacio(const t\_indice \*);

int indice\_lleno(const t\_indice \*, unsigned tamDato): indica si ya no se pueden añadir más registros

int indice\_buscar (const t\_indice\*, t\_reg\_indice\* reg\_indice): si el dni existe deja el registro en reg\_ind; int indice\_cargar (t\_indice\*, const char\* path): Carga el array desde un archivo

void indice\_vaciar(t\_indice\*): deja el índice en su estado de vacío.

# **TDA Indice (Array Redimensionable)**

Construir un TDA índice (t\_indice) en los archivos indice.h e indice.c, cuya estructura es un array donde cada elemento tiene la estructura {long dni; unsigned nro\_reg;} (t\_reg\_indice). Cada entrada del índice (dni-nro\_reg) se guarda en cada una de las posiciones del array.

Las primitivas del TDA son:

void indice\_crear ( t\_indice\*): toma memoria para 100 elementos e inicializa la estructura a índice vacío.

int indice\_insertar (t\_indice\*, const t\_reg\_indice\*): inserta en orden según el dni. Si no hay lugar en el array, toma memoria para un 30 % más de lo que tenía

int indice\_eliminar (t\_indice\*, t\_reg\_indice\*): elimina el dni informado. int indice\_vacio(const t\_indice \*);

int indice\_lleno(const t\_indice \*, unsigned tamDato): indica si ya no se pueden añadir más registros

int indice\_buscar (const t\_indice\*, t\_reg\_indice\* reg\_indice): si el dni existe deja el registro en reg\_ind; int indice\_cargar (t\_indice\*, const char\* path): Carga el array desde un archivo

void indice\_vaciar(t\_indice\*): libera la memoria utilizada por el índice.