Trabajo Práctico Número 3

Programación I – Laboratorio I

Tecnicatura Superior en Programación

UTN-FRA

Autor: Taboada Ezequiel.

Revisores: Scarafilo Germán, Cardozo Marina, Maggiori Gianni, Luchetta Giovanni.

Índice de contenido

Guía de contenidos

1. ENUNCIADO:

La FIFA requiere un programa que administre los jugadores de las 32 Selecciones de fútbol que participarán del mundial de Qatar 2022.
Cada selección no puede superar los 22 jugadores convocados.
Para esta tarea se deberá desarrollar una solución utilizando las funciones de la biblioteca **LinkedList**. Se deberán modelar las entidades que representan al jugador y la selección con todos sus datos asociados de tal manera que las mismas permitan interactuar con las estructuras ya definidas.

```
typedef struct
{
    int id;
    char nombreCompleto[100];
    int edad;
    char posicion[30];
    char nacionalidad[30];
    int idSeleccion;
    int isEmpty;
}Jugador;
```

```
typedef struct
{
    int id;
    char pais[30];
    char confederacion[30];
    int convocados;
    int isEmpty;
}Seleccion;
```

Nota: Se debe respetar el formato y tipos de datos.

2. MENÚ:

El sistema deberá tener el siguiente menú de opciones:

1. **CARGA DE ARCHIVOS:** Se deben leer los archivos .csv de jugadores y selecciones

2. **ALTA DE JUGADOR:** Se debe permitir ingresar un jugador calculando automáticamente el número de Id, el id de la selección debe quedar en 0. El resto de los campos se le pedirá al usuario.
   Nota: *el id del jugador debe ser autoincremental, único, autónomo y debe persistir, es decir, que su valor no debe reiniciarse cada vez que se reinicie la ejecución del programa, no debe depender de la posición de un array/lista ni calcularse en base a buscar el mayor id que tenga un jugador dentro del array/lista. El primer id disponible para la carga manual es el* ***371.***

3. **MODIFICACIÓN DE JUGADOR:** Se deberá mostrar la lista completa de jugadores con todos sus datos y se podrá elegir uno, permitiendo modificar **solamente**: nombre, edad, posición, nacionalidad.
   Este proceso debe contar con menú propio permitiendo elegir qué campo se desea modificar.

4. **BAJA DE JUGADOR:** Se deberá mostrar la lista completa de jugadores con todos sus datos, se podrá elegir uno y se eliminará el jugador del sistema.
   Nota: *si el jugador a dar de baja estaba convocado a una selección, en dicha selección se debe disminuir en 1 el contador de* ***convocados****.*

5. **LISTADOS:**
   A) TODOS LOS JUGADORES.
   B) TODAS LAS SELECCIONES.
   C) JUGADORES CONVOCADOS (únicamente)**.**
   Este proceso debe contar con menú propio permitiendo elegir qué listado se desea ver.

6. **CONVOCAR JUGADORES:**
   A) CONVOCAR: siempre y cuando el jugador no esté convocado en otra selección y la selección a donde será convocado no haya llegado a la cantidad máxima de convocados, se deberá asociar al jugador el id de la selección y en dicha selección se deberá aumentar el contador de **convocados** en 1. *Al momento de tener que elegir qué dato ingresar se deberá mostrar un listado con las opciones disponibles.*
   B) QUITAR DE LA SELECCIÓN: Se deberá mostrar el listado de jugadores convocados, se podrá elegir uno, se pondrá el id de selección en 0 dejando al jugador disponible para una nueva convocatoria y se deberá disminuir en 1 el contador de **convocados** de dicha selección.

7. **ORDENAR Y LISTAR:**
   - **A) JUGADORES POR NACIONALIDAD.**
   - **B) SELECCIONES POR CONFEDERACIÓN.**
   - **C) JUGADORES POR EDAD.**
   - **D) JUGADORES POR NOMBRE.**

   Este proceso debe contar con menú propio permitiendo elegir por cuál criterio se desea ordenar.

8. **GENERAR ARCHIVO BINARIO:** Generar y guardar en binario una nueva lista que contenga los jugadores convocados de una confederación ingresada por el usuario.

9. **CARGAR ARCHIVO BINARIO:** Se deberá leer e imprimir los datos del archivo generado en el punto 8.

10. **GUARDAR ARCHIVOS .CSV:** Se deberá guardar en sus respectivos archivos todos los cambios realizados en jugadores y selecciones.

11. **SALIR:** Termina la ejecución del programa previa confirmación del usuario, si se realizaron cambios en los archivos y estos no fueron guardados debería informarse antes de permitir la salida.

3. ACLARACIONES

Las funciones del LinkedList que deben utilizarse como mínimo son las siguientes**:**

LinkedList* ll_newLinkedList(void) *-> Crea y retorna un nuevo LinkedList. Es el constructor, ya que en él crearemos la estructura y daremos valores iniciales a los campos.*

void ll_deleteLinkedList(LinkedList* this) *-> Elimina el LinkedList*

int ll_isEmpty(LinkedList* this)*->retorna 0 o 1 para informar si el LinkedList está vacío o no .*

void ll_add(LinkedList* this, void* element) *->Agrega un elemento al final de LinkedList.*

void ll_remove(LinkedList* this, int index) *->Elimina un elemento en LinkedList, en el índice especificado.*

int ll_clear(LinkedList* this) *-> elimina los elementos del LinkedList, pero el LinkedList sigue existiendo.*

void* ll_get(LinkedList* this, int index) *->Retorna un puntero al elemento que se encuentra en el índice especificado.*

int ll_len(LinkedList* this) *->Retorna el tamaño del LinkedList.*

int ll_sort(LinkedList* this, int (*pFunc)(void* ,void*), int order)*->Ordena el LinkedList según el criterio pasado parámetro.*

Nota 1: *El programa debe evitar el acceso a las distintas opciones si no dispone de los datos necesarios.*

Nota 2: *Al momento de listar los jugadores, si está convocado se debe mostrar el nombre de dicha selección y si no el mensaje de no convocado.*

Nota 3: *Todos los campos de la estructura Selección serán de solo lectura (utilizar funciones getters), salvo el contador de convocados que debe ser de lectura y escritura (getters, setters).*

Nota 4: *Los operadores flecha (->) y/o punto (.) solo podrán usarse dentro de las funciones SETTERS, GETTERS y/o CONSTRUCTOR, en cualquier otra función o parte del proyecto no está permitido.*

Nota 5: *Se deberá contar con bibliotecas de funciones para entradas y salidas de datos, verificando y validando los mismos.*

4. Cómo realizar la entrega

El trabajo práctico deberá ser entregado en el repositorio de GIT **tps_Laboratorio_1** (el mismo que fue reportado para el TP_1), dentro de la carpeta **TP_3** , el proyecto en github deberá incluir los archivos .cproject y .project, caso contrario no se corregirá y contará como desaprobado.
Recordar que el repositorio **tps_Laboratorio_1** que contendrán todos los TPs deberá ser privado.
El mismo consistirá en el proyecto de Eclipse con el programa funcionando y comentado, respetando las reglas de estilo de la cátedra.
La compilación no deberá arrojar mensajes de error ni de warnings.