“jugadoras.h”

```
#ifndef JUGADORAS_H_
#define JUGADORAS_H_

#include <string>
#include <iostream>
#include <fstream>
using namespace std;

const int MAX_JUG = 100;

typedef struct tJugadora
{
	int id;
	string nombre;
	string apellido;
	int goles = 0;
};

typedef tJugadora* tJugadoraPtr;

typedef tJugadoraPtr tArrayJug[MAX_JUG];

typedef struct tJugadoras
{
	tArrayJug array_jugadoras;
	int cont = 0;
} ;

void cargarJugadoras(ifstream& archivo, tJugadoras& lista_jug);
void mostrarJugadoras(const tJugadoras& lista_jug);
bool buscarJugadora(int id, const tJugadoras& lista_jug, tJugadora &jug);

string getApellido(const tJugadora& jug);

void liberar_memoria(tJugadoras& lista_jugadoras);


#endif
```

“liga.h”

```
#ifndef LIGA_H_
#define LIGA_H_

#include <string>
#include<fstream>
#include "jugadoras.h"
using namespace std;

typedef int* tArrayJugEquipo; //array dinámico de identificadores de jugadoras
typedef struct tEquipo
{
	string nombre;
	int puntos = 0;
	int presupuesto;
	int num_jugadoras;
	tArrayJugEquipo jugadoras;
};


typedef struct tLiga
{
	int num_equipos=0;
	tEquipo equipos[10];
};

void cargarEquipos(ifstream& archivo, tLiga& liga);
void mostrarEquipos(const tLiga& liga,const tJugadoras& jugadoras);
tEquipo campeonLiga(ifstream& archivo, tLiga& liga);
void aumentarPresupuesto(tLiga& liga, string nombre);
bool ficharNuevaJugadora(tJugadoras& lista_jugadoras, tLiga& liga, string equipo,
	int id_jug, string nombre_jug, string apellido_jug, int goles_jug);
void descensoEquipo(string eq, tLiga& liga, tJugadoras& jugadoras);


string getNombre(const tEquipo& equipo);
int getPuntos(const tEquipo& equipo);
void liberar_memoria(tLiga& liga);

#endif
```

“main.cpp”

```
#include <iostream>
#include <fstream>
#include <string>
#include "jugadoras.h"
#include "liga.h"
using namespace std;

void fichajes(ifstream& archivo, tJugadoras& lista_jugadoras, tLiga liga);

int main()
{
	tJugadoras lista_jugadoras;
	tLiga liga;
	ifstream archivo;
	tEquipo equipo_campeon;
	_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);

	archivo.open("datos.txt");
	bool abierto = archivo.is_open();
	if (abierto)
	{
		cargarJugadoras(archivo, lista_jugadoras);
		mostrarJugadoras(lista_jugadoras);
		cargarEquipos(archivo, liga);
		mostrarEquipos(liga, lista_jugadoras);

		//Equipo campeón

		equipo_campeon = campeonLiga(archivo, liga);
		cout << "El campeón de la liga es: " << getNombre(equipo_campeon)
			<< " con " << getPuntos(equipo_campeon) << " puntos" <<
			endl <<endl;

		//Aumentar presupuesto

		string equipo;
		archivo >> equipo;
		aumentarPresupuesto(liga, equipo);
		mostrarEquipos(liga, lista_jugadoras);
```

```
		//Fichajes
		fichajes(archivo, lista_jugadoras, liga);
		mostrarJugadoras(lista_jugadoras);
		mostrarEquipos(liga, lista_jugadoras);

		//Descenso equipo

		string descenso;
		archivo >> descenso;
		descensoEquipo(descenso, liga, lista_jugadoras);
		cout << "Desciende el equipo " << descenso << endl << endl;
		mostrarJugadoras(lista_jugadoras);
		mostrarEquipos(liga, lista_jugadoras);

		cout << "Se libera memoria dinámica" << endl;
		liberar_memoria(liga);
		liberar_memoria(lista_jugadoras);
		archivo.close();

	}
	else
		cout << "Archivo no abierto" << endl;
	return 0;
}

void fichajes(ifstream& archivo, tJugadoras& lista_jugadoras, tLiga liga)
{
	int num_fichajes;
	int id_jug, goles_jug;
	string equipo, nombre_jug, apellido_jug;
	tJugadoraPtr jug;

	archivo >> num_fichajes;
	for (int i = 0; i < num_fichajes; i++)
	{
		archivo >> equipo;
		archivo >> id_jug >> nombre_jug >> apellido_jug >> goles_jug;
		bool fichada = ficharNuevaJugadora(lista_jugadoras, liga, equipo,
			id_jug, nombre_jug, apellido_jug, goles_jug);
		if (!fichada)
			cout << equipo << " no puede fichar más jugadoras" << endl;
	}
	cout << endl;
}
```

“liga.cpp”

```
#include "liga.h"
#include "jugadoras.h"

void cargarEquipo(ifstream& archivo, tEquipo& equipo);
void mostrarEquiposAux(const tLiga& liga, const tJugadoras &jugadoras, int num);
void mostrarEquipo(const tEquipo& equipo, const tJugadoras &jugadoras);

int sumarPuntos(string eq, tLiga& liga, int puntos);
int posicionEquipo(tLiga& liga, string nombre);
void actualizarCampeon(string eq, string& campeon, int& max_puntos, int p);

void cargarEquipos(ifstream& archivo, tLiga& liga)
{
	int num_equipos;
	archivo >> num_equipos;
	for (int i = 0; i < num_equipos; i++)
	{
		cargarEquipo(archivo, liga.equipos[i]);
		liga.num_equipos++;
	}
}

void cargarEquipo(ifstream& archivo, tEquipo& equipo)
{
	archivo >> equipo.nombre;
	archivo >> equipo.presupuesto;
	archivo >> equipo.num_jugadoras;
	 equipo.jugadoras = new int[equipo.presupuesto * 3 / 1000];
	 for (int i = 0; i < equipo.num_jugadoras; i++)
		archivo >> equipo.jugadoras[i];
}

void mostrarEquipos(const tLiga& liga, const tJugadoras &jugadoras)
{
	cout << "- - - - -" << endl;
	cout << "EQUIPOS" << endl;
	cout << "- - - - -" << endl;
	mostrarEquiposAux(liga, jugadoras, 0);
	cout << endl;
}
```

```
void mostrarEquiposAux(const tLiga& liga, const tJugadoras & jugadoras, int pos)
{
	if (pos < liga.num_equipos)
	{
		mostrarEquipo(liga.equipos[pos], jugadoras);
		mostrarEquiposAux(liga, jugadoras, pos + 1);
	}
}

void mostrarEquipo(const tEquipo& equipo, const tJugadoras &jugadoras)
{
	tJugadora jug;
	cout << "Nombre: " << equipo.nombre << ","
	     << " Presupuesto: " << equipo.presupuesto << ","
	     << " Puntos: " << equipo.puntos << ","
	     << " Jugadoras: " << equipo.num_jugadoras << endl;
	if (equipo.num_jugadoras > 0)
		cout << "Plantilla: ";
	for (int i = 0; i < equipo.num_jugadoras; i++)
		if (buscarJugadora(equipo.jugadoras[i], jugadoras, jug))
			cout << getApellido(jug) << ", ";
	 cout << endl << endl;
}
```

```
tEquipo campeonLiga(ifstream& archivo, tLiga& liga)
{
	string eq1, eq2;
	int pt1, pt2, p;
	int num_enfrentamientos;
	int max_puntos = 0;
	string campeon = "";

	archivo >> num_enfrentamientos;
	for (int i = 0; i < num_enfrentamientos; i++)
	{
		archivo >> eq1 >> pt1 >> eq2 >> pt2;
		if (pt1 == pt2)
		{
			p = sumarPuntos(eq1, liga, 1);
			actualizarCampeon(eq1, campeon, max_puntos, p);
			p = sumarPuntos(eq2, liga, 1);
			actualizarCampeon(eq2, campeon, max_puntos, p);
		}

		else if (pt1 > pt2)
		{
			p = sumarPuntos(eq1, liga, 3);
			actualizarCampeon(eq1, campeon, max_puntos, p);
		 }
		else
		{
			p = sumarPuntos(eq2, liga, 3);
			 actualizarCampeon(eq2, campeon, max_puntos, p);
		}
	}
	int pos = posicionEquipo(liga, campeon);
	return liga.equipos[pos];
}

int sumarPuntos(string eq, tLiga& liga, int puntos)
{
	int pos = posicionEquipo(liga, eq);
	liga.equipos[pos].puntos += puntos;
	return liga.equipos[pos].puntos;
}
```

```
int posicionEquipo(tLiga& liga, string nombre)
{
	int pos = -1;
	bool encontrado = false;
	int i = 0;
	while (i < liga.num_equipos && liga.equipos[i].nombre != nombre)
		i++;
	if (i < liga.num_equipos)
		pos = i;
	return pos;
}

void actualizarCampeon(string eq, string &campeon, int &max_puntos, int p)
{
	if (p > max_puntos)
	{
		max_puntos = p;
		campeon = eq;
	}
}


void aumentarPresupuesto(tLiga& liga, string id)
{
	int pos = posicionEquipo(liga, id);
	if (pos != -1)
	{
		liga.equipos[pos].presupuesto += 1000;
		int* aux = new int[(liga.equipos[pos].presupuesto * 3 / 1000)];
		for (int i = 0; i < liga.equipos[pos].num_jugadoras; i++)
			aux[i] = liga.equipos[pos].jugadoras[i];
		delete[] liga.equipos[pos].jugadoras;
		liga.equipos[pos].jugadoras = aux;
		cout << "Se aumenta el presupuesto de " << id << endl;
	}
}
```

```
bool ficharNuevaJugadora(tJugadoras& lista_jugadoras, tLiga& liga, string equipo,
                         int id_jug, string nombre_jug, string apellido_jug, int goles_jug)
{
        bool insertada = false;
        int pos = posicionEquipo(liga, equipo);
        if (pos != -1 && liga.equipos[pos].num_jugadoras <
                              liga.equipos[pos].presupuesto * 3 / 1000)
        {
                insertada = true;
                liga.equipos[pos].jugadoras[liga.equipos[pos].num_jugadoras] = id_jug;
                liga.equipos[pos].num_jugadoras++;
                tJugadoraPtr jug;
                crearJugadora(jug, id_jug, nombre_jug, apellido_jug, goles_jug);
                insertarJugadora(lista_jugadoras, jug);
        }
        return insertada;
}

void descensoEquipo(string eq, tLiga& liga, tJugadoras& jugadoras)
{
        int pos = posicionEquipo(liga, eq);
        if (pos != -1)
        {
                tEquipo equipo = liga.equipos[pos];
                for (int j = pos; j < liga.num_equipos; j++)
                        liga.equipos[j] = liga.equipos[j + 1];
                liga.num_equipos--;
                 for (int i = 0; i < equipo.num_jugadoras; i++)
                          eliminarJugadora(jugadoras, equipo.jugadoras[i]);
                 delete[] equipo.jugadoras;
                 equipo.jugadoras = nullptr;
        }
}

string getNombre(const tEquipo& equipo)
{
        return equipo.nombre;
}

int getPuntos(const tEquipo& equipo)
{
         return equipo.puntos;
}
```

```
void liberar_memoria(tLiga& liga)
{
	for (int i = 0; i < liga.num_equipos; i++)
	{
		delete[] liga.equipos[i].jugadoras;
		liga.equipos[i].jugadoras = nullptr;
	 }

}
```

“jugadoras.cpp”

```
#include "jugadoras.h"
#include <fstream>
using namespace std;

void cargarJugadora(ifstream& archivo, tJugadora& lista_jug);
void mostrarJugadoras_aux(const tJugadoras& lista, int pos);
void mostrarJugadora(const tJugadora& jug);

void cargarJugadoras(ifstream& archivo, tJugadoras& lista_jug)
{
	int num;

	archivo >> num;
	for (int i = 0; i < num;i++)
	{
		tJugadora* ptr = new tJugadora;
		cargarJugadora(archivo, *ptr);
		lista_jug.array_jugadoras[lista_jug.cont] = ptr;
		lista_jug.cont++;
	}
}

void cargarJugadora(ifstream & archivo, tJugadora& jug)
{
	archivo >> jug.id >> jug.nombre >> jug.apellido >> jug.goles;
}

void mostrarJugadoras(const tJugadoras& lista_jug)
{
	cout << "- - - - -"<<endl;
	cout <<"JUGADORAS"<<endl;
	cout <<"- - - - -"<<endl;
	mostrarJugadoras_aux(lista_jug,0);
}
```

```
void mostrarJugadoras_aux(const tJugadoras& lista_jug, int pos)
{
	if (pos < lista_jug.cont)
	{
		mostrarJugadora(*lista_jug.array_jugadoras[pos]);
		cout << endl;
		mostrarJugadoras_aux(lista_jug, pos + 1);
	}
}

void mostrarJugadora(const tJugadora& jug)
{
	cout << "Nombre: " << jug.nombre << " " << jug.apellido << ", Goles: " <<
jug.goles;
}

bool buscarJugadora(int id, const tJugadoras& lista_jug, tJugadora& jug)
{
	int i = 0;
	bool encont;
	while (i < lista_jug.cont && lista_jug.array_jugadoras[i]->id != id)
		i++;
	encont = i < lista_jug.cont;
	 if (encont)
		jug = *lista_jug.array_jugadoras[i];
	 return encont;
}

string getApellido(const tJugadora& jug)
{
	return jug.apellido;
}

void liberar_memoria(tJugadoras& lista_jugadoras)
{
	for (int i = 0; i < lista_jugadoras.cont; i++)
	{
		delete lista_jugadoras.array_jugadoras[i];
		lista_jugadoras.array_jugadoras[i] = nullptr;
	}
}
```