Enunciado Trabajo Práctico

Gran DT

Algoritmos y Estructuras de Datos

Ing. Diego Azcurra – Ing. Damián Santos

UNLA

2017

|  |  |  |
| --- | --- | --- |
| Padrón | Nombre | Evaluación Individual |
| 39281276 | Gonzalo Cerbelli |  |
| 38469373 | Agustín Fontán |  |
| 39105048 | Juan Pablo Federico |  |
| 33691081 | Axel Diorno |  |
| 39461683 | Andrés Filoso |  |
|  |  |  |
| Evaluación Trabajo |  | |

## Objetivo

Desarrollar una aplicación en C/C++ que permita administrar el juego del Gran DT.

## Entidades

## Club de Fútbol:

Los clubes son los equipos que participan del Torneo.

Se identifican por:

* Id: número entero.
* Nombre: Cadena de caracteres.

## Jugadores

Los jugadores pertenecen a los equipos.

Se identifican por

* Id: número entero.
* Nombre: Cadena de caracteres.
* Posición: enumerado: ARQ, DEF, MED, DEL.
* Valor: Precio del jugador. Float.

## Equipos

Los equipos son armados por los usuarios del Gran DT.

Están definidos por:

* Id: número entero.
* Nombre: nombre del equipo. Cadena.
* NombreDelUsuario: nombre del usuario que armó el equipo. Cadena.
* Lista de arqueros.
* Lista de Defensores.
* Lista de Mediocampistas.
* Lista de Delanteros

## Puntajes de Fecha

Define los puntos que sumó cada jugador.

Se determina por:

* Id jugado: número entero.
* Fecha: número de fecha.
* Puntos: número entero (pueden ser negativos).

## Archivos.

## Archivo de Configuración

Se leerán los parámetros de configuración desde un archivo denominado configuracion.conf, con el siguiente formato.

NombreDelParametro=ValorDelParametro

Las variantes son:

* ValorMaximoXEquipo. Representa el valor máximo que podrá tener cada equipo. Dato tipo float.
* ARQ: Representa la cantidad máxima de arqueros que pueden tener los equipos. Dato entero.
* DEF: Representa la cantidad máxima de defensores que pueden tener los equipos. Dato entero.
* MED: Representa la cantidad máxima de mediocampistas que pueden tener los equipos. Dato entero.
* DEL: Representa la cantidad máxima de delanteros que pueden tener los equipos. Dato entero.

Por ejemplo:

ValorMaximoXEquipo=100000000

ARQ=3

DEF=5

MED=5

DEL=4

## Clubes.db

Posee los campos:

* Id: número entero.
* Nombre: Cadena de caracteres.

*Ejemplo*

1;Racing

2;River

3;San Lorenzo

….

## Jugadores.db

Posee los campos:

* Id: número entero.
* Nombre: Cadena de caracteres.
* idClub: número entero, clave en archivo clubes.db.
* Posición: enumerado: ARQ, DEF, MED, DEL.
* Valor: Precio del jugador. Float.

*Ejemplo*

1;Gustavo Bou;1;DEL;5000000

2;Lisandro López;1;DEL;5000000

….

## Equipos.db

Posee los campos:

* Id: número entero.
* Nombre: nombre del equipo. Cadena.
* NombreDelUsuario: nombre del usuario que armó el equipo. Cadena.

*Ejemplo*

1;UNLA DreamTeam;Damian Santos

2;La naranja mecánica;Diego Azcurra

…

## JugadoresPorEquipo.db

Posee los campos:

* idEquipo: número Entero. Clave en Equipos.db.
* idJugador: número Entero. Clave en Jugadores.db.

*Ejemplo*

1;1

1;2

2;1

…

## PuntosPorJugador.db

Posee los campos:

* idJugador: número entero, clave en jugadores.db
* fecha: número entero que identifica la fecha del torneo (1, 2, 3….)
* puntos: número entero, puede ser negativo.

*Ejemplo*

1;1;10

3;1;4

2;1;5

…

NOTA:

* Como parte del trabajo, los alumnos deberán desarrollar un archivo de pruebas con datos válidos para ejecutar las validaciones.
* En todos los casos, el carácter de separación de los campos de cada registro de los archivos será el “;”. Al final de cada registro no habrá “;”, sólo un salto de línea.
* Ninguno de los archivos está ordenado. Sólo el de PuntosPorJugador.db que está ordenado por fecha.
* Los archivos no tendrán errores de formato ni de datos.

## Requerimientos

Se pide desarrollar una aplicación que permita:

1. Procesamiento de archivos
   1. Configuraciones.

Al iniciar el programa, se deberán cargar (en forma automática, es decir, sin intervención del usuario) los parámetros de operación desde el archivo configuracion.conf.

* 1. Archivos de Datos.

Deberá existir una opción para la carga de cada uno de los archivos de datos:

* clubes.db
* jugadores.db
* equipos.db
* jugadoresPorEquipo.db

Se deberá validar que:

* Los jugadores pertenezcan a clubes existentes.
* Ningún equipo viole alguno de los parámetros definidos en el archivo de configuración, por ejemplo, que el valor total supere al máximo o que haya más arqueros que los permitidos. Si este fuera el caso, ese equipo no será tenido en cuenta para el resto de la aplicación.

Se deberá guardar un registro de los errores encontrados en un archivo errores.txt con el siguiente formato:

* Número de error.
* Descripción.

1. Procesamiento de fechas:

Esta opción permitirá procesar los puntos de los jugadores en cada fecha dese el archivo “puntosPorJugador.db”. Se procesará el archivo en su totalidad, es decir, todas las fechas para las que existan datos. Al estar ordenado por fecha, el sistema procesará fecha por fecha y al final de cada una, emitirá un archivo con las siguientes características:

El nombre será puntajesFechaXX.txt donde XX representa el número de la fecha procesada. Deberá estar ordenado por la cantidad de puntos alcanzados por cada equipo en esa fecha en forma descendente.

Campos:

* Posición (número secuencial de 1 a n)
* Id del equipo
* Nombre Equipo
* Nombre Jugador
* Puntos

Al finalizar la lectura del archivo de puntosPorJugador, se emitirán dos archivos:

* puntajesTotalesPorEquipo.txt: Mostrará el total de puntos de cada equipo como la sumatoria de todas las fechas. Deberá estar ordenado desde el equipo con más puntos al de menor cantidad de puntos.

Campos:

* Posición (número secuencial de 1 a n)
* Id del equipo
* Nombre Equipo
* Nombre Jugador
* Puntos
* puntajesPorJugador.txt: Mostrará el ranking de puntos totales ganados por cada jugador en todas las fechas. El orden es de mayor a menor puntaje.

Campos:

* Posición (número secuencial de 1 a n)
* Id de jugador
* puntos

Nota:

Los campos de todos los archivos deberán estar separados por “;”.

## Presentación

A continuación se detalla el cronograma de actividades para el trabajo práctico:

12/05/17: Presentación del enunciado y armado de grupos.

19/05/17: **Primer Entrega:** “Estrategia de Resolución”. Se presentará la estrategia de resolución propuesta por el equipo.

26/05/17: Seguimiento del TP.

02/06/17: Seguimiento del TP.

09/06/17: No hay actividades previstas para el TP por ser fecha de parcial.

16/06/17: **Segunda Entrega:** Presentación y Evaluación del TP.

23/06/17: No hay actividades previstas para el TP por ser fecha de recuperatorio parcial.

30/6/17: **Recuperatorio del TP**.

**Primer Entrega:** “Estrategia de Resolución”.

Para esta entrega deberá presentarse un documento impreso (en un folio o carpeta) conteniendo, como mínimo:

* Carátula de presentación con los datos de los integrantes del equipo (primer hoja de este documento).
* Indice de contenidos (en caso de ser necesario).
* Desarrollo de la estrategia de resolución detallando:
  + Estructuras a utilizar (arrays, pilas, listas, colas).
  + TDAs y sus relaciones (diagrama de interacción de todos los componentes).
  + Estrategia de resolución de operaciones.
* División del trabajo y cronograma: se deberá presentar la división de actividades dentro del grupo y un cronograma de trabajo a alto nivel.

Todas las hojas deben estar numeradas.

**Segunda Entrega**.

Para esta entrega deberá presentarse:

* Una copia impresa del enunciado del trabajo práctico (TODO este documento, incluyendo los anexos).
* Una copia impresa de la estrategia de resolución final del trabajo práctico (con los mismos requerimientos que en la “Primer Entrega”.
* Una copia impresa de todos los archivos de prueba presentados por los alumnos.
* Una copia impresa de todos los archivos del proyecto (.h y .cpp). Poner como encabezado de cada hoja el nombre del archivo.
* Un CD conteniendo en formato digital todos los puntos anteriores y el proyecto completo.
* Se deberá incluir la primer entrega corregida.

La presentación deberá ser en un folio o carpeta, en forma prolija y debidamente identificada. Los CDs deberán contener el número de grupo y el nombre y los padrones o documentos de identidad de cada uno de los integrantes y deberán estar correctamente adjuntos al resto del trabajo práctico de forma tal que no puedan perderse. Además, deberá incluirse **todo** el proyecto desarrollado (**la carpeta completa** generada por el IDE, con los archivos del proyecto y el código fuente) incluyendo los archivos de pruebas.

Todas las hojas deben estar numeradas.

El incumplimiento de cualquiera de las normas de entrega implicará la desaprobación del trabajo práctico.

Metodología de evaluación:

La Evaluación de los trabajos prácticos contará con una etapa grupal y una individual.

* Grupal: Se realizará un conjunto de pruebas sobre el trabajo presentado por los alumnos en presencia de los mismos. Se deberá aprobar la totalidad de las pruebas. En caso de que una prueba falle, los alumnos podrán intentar corregir el código mientras dure la evaluación.
* Individual: Se realizará una evaluación individual oral o escrita para cada alumno. Los temas a evaluar podrán ser, por ejemplo: preguntas teóricas sobre el contenido de la materia, preguntas sobre el trabajo práctico, codificación de alguna primitiva o modificación del trabajo práctico, etc.

La nota final del trabajo se calculará en función de las notas obtenidas en forma grupal e individual. La nota grupal será el promedio entre la primer presentación y el recuperatorio (en caso de necesitarlo). Por este motivo, SOLO deberán presentarse aquellos grupos que hayan concluido TODO el trabajo práctico ya que no se harán evaluaciones parciales.

## Revisiones

2017-05-21: Se cambiaron los formatos de los números en los archivos para que no tengan puntos.

Anexo – Correcciones

|  |  |  |
| --- | --- | --- |
| Prueba | Resultado | Comentario |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Anexo – Correcciones (copia para los alumnos)

|  |  |  |
| --- | --- | --- |
| Prueba | Resultado | Comentario |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Estrategia de resolución Inicial.**

**Entidades:** El planteo de las mismas se hará de la siguiente manera: En las entidades crearemos los TDA llamados Club, Jugador, Equipo y PuntosFecha.

Además crearemos la lista de clubes, la de jugadores y la de equipos que además tendrá como atributo 4 listas con los diferentes jugadores según cada rol (lista de arqueros, defensores, mediocampistas y delanteros). Por otro lado utilizaremos una pila para los puntajes de cada jugador en cada fecha.

**Archivos:**

* Configuracion.conf: se utilizarán los valores límite (máximos) que contiene para hacer las validaciones a la hora de insertar los registros de los otros archivos en sus TDA y sus listas correspondientes.
* Clubes.db: se cargará cada uno de sus registros en el TDA Club.
* Jugadores.db: se cargará cada uno de sus registros en el TDA Jugador.
* Equipos.db: se cargará cada uno de sus registros en el TDA Equipo.
* JugadoresPorEquipo.db: En cada registro se cargará un jugador en un TDA Equipo (según su id equipo), en una de sus 4 listas según corresponda su posición (ARQ,DEF,MED,DEL).
* PuntosPorJugador.db: Se cargará cada uno de sus registros en el TDA PuntosFecha que posteriormente se incluirá en la pila de dicho tipo.

**Distribución de tareas:** Para realizar el TP cuatrimestral nos dividiremos en dos sub-grupos de 3 personas cada uno. Cada uno de ellos tendrá las siguientes funciones:

* Sub-grupo 1: Creación de los TDA, adaptación de las listas y pilas según se requieran y carga de los archivos que contienen los datos para así poder realizar las pruebas.
* Sub-grupo 2: Desarrollo del main.cpp que incluye las instancias de los TDAs, listas y pilas, la lectura correcta de los archivos para su posterior carga en las estructuras de datos correspondientes y las validaciones de la carga en éstas últimas.

**Estrategia de resolución Final.**

Se utilizaron las mismas estructuras mencionadas en la estrategia inicial, con excepción de la pila de puntosFecha, la cual se reemplazó por una cola, ya que nos pareció más óptima a la hora de ordenar y mostrar los datos contenidos en ella.

Para la implementación del proyecto utilizamos el sistema de control de versiones Git y creamos un repositorio en GitHub alojado en la siguiente URL:

[**https://github.com/andresfiloso/bigdt.git**](https://github.com/andresfiloso/bigdt.git)

#ifndef CLUB\_H\_INCLUDED

#define CLUB\_H\_INCLUDED

#include <string>

using namespace std;

typedef struct {

int idClub;

string nombre;

} Club;

void constructor(Club &club);

void destructor(Club &club);

void setIdClub(Club &club, int idClub);

int getIdClub(Club &club);

string getNombre(Club &club);

void setNombre(Club &club, string nombre);

#endif // CLUB\_H\_INCLUDED

#ifndef EQUIPO\_H\_INCLUDED

#define EQUIPO\_H\_INCLUDED

#include <string>

#include "ListaJugadores.h"

using namespace std;

typedef struct {

int idEquipo;

string nombreEquipo;

string nombreUsuario;

float monto;

ListaJ ARQ;

ListaJ DEF;

ListaJ MED;

ListaJ DEL;

} Equipo;

void constructor(Equipo &equipo);

void destructor(Equipo &equipo);

void setIdEquipo(Equipo &equipo, int idEquipo);

int getIdEquipo(Equipo &equipo);

string getNombreEquipo(Equipo &equipo);

void setNombreEquipo(Equipo &equipo, string nombreEquipo);

string getNombreUsuario(Equipo &equipo);

void setNombreUsuario(Equipo &equipo, string nombreUsuario);

float getMonto(Equipo &equipo);

void setMonto(Equipo &equipo, float monto);

ListaJ& getListaARQ(Equipo &equipo);

void setListaARQ(Equipo &equipo, ListaJ &ARQ);

ListaJ& getListaDEF(Equipo &equipo);

void setListaDEF(Equipo &equipo, ListaJ &DEF);

ListaJ& getListaMED(Equipo &equipo);

void setListaMED(Equipo &equipo, ListaJ &MED);

ListaJ& getListaDEL(Equipo &equipo);

void setListaDEL(Equipo &equipo, ListaJ &DEL);

#endif // EQUIPO\_H\_INCLUDED

#ifndef JUGADOR\_H\_INCLUDED

#define JUGADOR\_H\_INCLUDED

#include <string>

using namespace std;

enum Posicion{

INDEF,

ARQ,

DEF,

MED,

DEL

};

typedef struct {

int idJugador;

string nombre;

Posicion posicion;

float valor;

} Jugador;

void constructor(Jugador &jugador);

void destructor(Jugador &jugador);

void setIdJugador(Jugador &jugador, int idJugador);

int getIdJugador(Jugador &jugador);

string getNombre(Jugador &jugador);

void setNombre(Jugador &jugador, string nombre);

Posicion& getPosicion(Jugador &jugador);

void setPosicion(Jugador &jugador, Posicion posicion);

float getValor(Jugador &jugador);

void setValor(Jugador &jugador, float valor);

#endif // CLUB\_H\_INCLUDED

#ifndef PUNTOSFECHA\_H\_INCLUDED

#define PUNTOSFECHA\_H\_INCLUDED

typedef struct {

int idJugador;

int fecha;

int puntos;

} PuntosFecha;

void constructor(PuntosFecha &puntosFecha);

void destructor(PuntosFecha &puntosFecha);

void setIdJugador(PuntosFecha &puntosFecha, int idJugador);

int getIdJugador(PuntosFecha &puntosFecha);

void setFecha(PuntosFecha &puntosFecha, int fecha);

int getFecha(PuntosFecha &puntosFecha);

int getPuntos(PuntosFecha &puntosFecha);

#endif // EQUIPO\_H\_INCLUDED

#include "Club.h"

void constructor(Club &club){

club.idClub = 0;

club.nombre = "";

}

void destructor(Club &club){

}

int getIdClub(Club &club){

return club.idClub;

}

void setIdClub(Club &club, int idClub){

club.idClub = idClub;

}

string getNombre(Club &club){

return club.nombre;

}

void setNombre(Club &club, string nombre ){

club.nombre = nombre;

}

#include "Equipo.h"

#include <string>

#include "ListaEquipos.h"

using namespace std;

void constructor(Equipo &equipo){

equipo.idEquipo = 0;

equipo.nombreEquipo = "";

equipo.nombreUsuario = "";

equipo.monto = 0;

crearLista(equipo.ARQ);

crearLista(equipo.DEF);

crearLista(equipo.MED);

crearLista(equipo.DEL);

}

void destructor(Equipo &equipo){}

int getIdEquipo(Equipo &equipo){

return equipo.idEquipo;

}

void setIdEquipo(Equipo &equipo, int idEquipo){

equipo.idEquipo = idEquipo;

}

string getNombreEquipo(Equipo &equipo){

return equipo.nombreEquipo;

}

void setNombreEquipo(Equipo &equipo, string nombreEquipo){

equipo.nombreEquipo = nombreEquipo;

}

string getNombreUsuario(Equipo &equipo){

return equipo.nombreUsuario;

}

void setNombreUsuario(Equipo &equipo, string nombreUsuario){

equipo.nombreUsuario = nombreUsuario;

}

float getMonto(Equipo &equipo){

return equipo.monto;

}

void setMonto(Equipo &equipo, float monto){

equipo.monto = monto;

}

ListaJ& getListaARQ(Equipo &equipo){

return equipo.ARQ;

}

void setListaARQ(Equipo &equipo, ListaJ &ARQ){

equipo.ARQ = ARQ;

}

ListaJ& getListaDEF(Equipo &equipo){

return equipo.DEF;

}

void setListaDEF(Equipo &equipo, ListaJ &DEF){

equipo.DEF = DEF;

}

ListaJ& getListaMED(Equipo &equipo){

return equipo.MED;

}

void setListaMED(Equipo &equipo, ListaJ &MED){

equipo.MED = MED;

}

ListaJ& getListaDEL(Equipo &equipo){

return equipo.DEL;

}

void setListaDEL(Equipo &equipo, ListaJ &DEL){

equipo.DEL = DEL;

}

#include <string>

#include "Jugador.h"

void constructor(Jugador &jugador){

jugador.idJugador = 0;

jugador.posicion = INDEF;

jugador.nombre = "";

jugador.valor = 0;

}

void destructor(Jugador &jugador){}

void setIdJugador(Jugador &jugador, int idJugador){

jugador.idJugador = idJugador;

}

int getIdJugador(Jugador &jugador){

return jugador.idJugador;

}

string getNombre(Jugador &jugador){

return jugador.nombre;

}

void setNombre(Jugador &jugador, string nombre){

jugador.nombre = nombre;

}

Posicion& getPosicion(Jugador &jugador){

return jugador.posicion;

}

void setPosicion(Jugador &jugador, Posicion posicion){

jugador.posicion = posicion;

}

float getValor(Jugador &jugador){

return jugador.valor;

}

void setValor(Jugador &jugador, float valor){

jugador.valor = valor;

}

#include "PuntosFecha.h"

void constructor(PuntosFecha &puntosFecha){

int idJugador = 0;

int fecha = 0;

int puntos = -100;

}

void destructor(PuntosFecha &puntosFecha){}

void setIdJugador(PuntosFecha &puntosFecha, int idJugador){

puntosFecha.idJugador = idJugador;

}

int getIdJugador(PuntosFecha &puntosFecha){

return puntosFecha.idJugador;

}

void setFecha(PuntosFecha &puntosFecha, int fecha){

puntosFecha.fecha = fecha;

}

int getFecha(PuntosFecha &puntosFecha){

return puntosFecha.fecha;

}

void setPuntos(PuntosFecha &puntosFecha, int puntos){

puntosFecha.puntos = puntos;

}

int getPuntos(PuntosFecha &puntosFecha){

return puntosFecha.puntos;

}

**Programa principal:**

struct Configuracion{

float ValorMaximoXEquipo;

int maxArq;

int maxDef;

int maxMed;

int maxDel;

};

struct PuntosEquipo{

int idEquipo;

string nomEquipo;

string nomUsuario;

int fecha;

int puntos;

};

Configuracion cargarConfig(FILE \*pf);

void cargarClubes(ListaC &listaClubes, FILE \*pf, FILE \*errores);

void cargarJugadores(ListaJ &listaJugadores, ListaC &listaClubes, FILE \*pf, FILE \*errores);

void cargarEquipos(ListaE &listaEquipos, FILE \*pf, FILE \*errores);

void cargarJugadoresPorEquipo(ListaE &listaEquipos, ListaJ &listaJugadores, FILE \*pf, FILE \*errores, Configuracion config);

void cargarPuntosPorJugador(ListaE &listaEquipos, ListaJ &listaJugadores, Cola &colaPuntosFecha, FILE \*pf, FILE \*errores);

// METODOS DE PRUEBA

bool traerClubes(ListaC &listaClubes);

bool traerJugadores(ListaJ &listaJugadores);

bool traerEquipos(ListaE &listaEquipos);

bool traerJugadorePorEquipo(ListaE &listaEquipos);

// METODOS DE VALIDACION

bool validarJugador(Equipo &equipo, Configuracion config, FILE \*errores, Posicion posicion);

bool validarValorEquipo(Equipo &equipo, Configuracion config, FILE \*errores);

// METODO DE ORDENAMIENTO E IMPRESION DE ARCHIVOS

void imprimirOrdenado(PuntosEquipo puntosEquipo[], int ultimaFecha, ListaE &listaEquipos, ListaJ &listaJugadores, Cola &colaPuntosFecha);

int main()

{

setlocale(LC\_ALL, "");

ListaC listaClubes;

crearLista(listaClubes);

ListaJ listaJugadores;

crearLista(listaJugadores);

ListaE listaEquipos;

crearLista(listaEquipos);

Cola colaPuntosFecha;

crearCola(colaPuntosFecha);

Configuracion config;

FILE \*errores;

FILE \*pf;

config = cargarConfig(pf);

int opcion = 0;

while(opcion!=6){

system("cls");

cout << "Ingrese el archivo del que desea cargar los datos: " << endl;

cout << "1. Clubes.db" << endl;

cout << "2. Jugadores.db" << endl;

cout << "3. Equipos.db" << endl;

cout << "4. JugadoresPorEquipo.db" << endl;

cout << "5. PuntosPorJugador.db" << endl;

cout << "6. Salir" << endl;

cin >> opcion;

switch(opcion){

case 1:

cargarClubes(listaClubes, pf, errores); //TERMINADO

break;

case 2:

cargarJugadores(listaJugadores, listaClubes, pf, errores); //TERMINADO

break;

case 3:

cargarEquipos(listaEquipos, pf, errores); // TERMINADO

break;

case 4:

cargarJugadoresPorEquipo(listaEquipos, listaJugadores, pf, errores, config);

break;

case 5:

cargarPuntosPorJugador(listaEquipos, listaJugadores, colaPuntosFecha, pf, errores);

break;

case 6:

break;

case 11:

traerClubes(listaClubes);

break;

case 12:

traerJugadores(listaJugadores);

break;

case 13:

traerEquipos(listaEquipos);

break;

case 14:

traerJugadorePorEquipo(listaEquipos);

break;

}

system("PAUSE");

}

eliminarLista(listaClubes);

eliminarLista(listaJugadores);

eliminarLista(listaEquipos);

eliminarCola(colaPuntosFecha);

return 0;

}

**Procedimientos y funciones llamados por el programa principal:**

**Configuracion cargarConfig(FILE \*pf)**{

Configuracion aux;

if ((pf = fopen("configuracion.conf", "r")) == NULL){

cout << "Error al abrir el archivo de configuracion" << endl;

Sleep(2000);

exit(0);

}

fscanf(pf, "ValorMaximoXEquipo=%f\n", &aux.ValorMaximoXEquipo);

fscanf(pf, "ARQ=%d\n", &aux.maxArq);

fscanf(pf, "DEF=%d\n", &aux.maxDef);

fscanf(pf, "MED=%d\n", &aux.maxMed);

fscanf(pf, "DEL=%d\n", &aux.maxDel);

fclose(pf);

return aux;

}

**void cargarClubes(ListaC &listaClubes, FILE \*pf, FILE \*errores)**{

Club club;

constructor(club);

int xIdClub;

char xNombre[60];

if ((errores = fopen("errores.txt", "a")) == NULL){

cout << "Error al abrir el archivo errores.txt" << endl;

Sleep(2000);

exit(0);

}

if ((pf = fopen("Clubes.db", "r")) == NULL){

cout << "Error al abrir el archivo Clubes.db" << endl;

Sleep(2000);

exit(0);

}

while(!feof(pf)){

fscanf(pf, "%d;%[A-Z a-z 0-9 .' ä-ü á-ú]\n", &xIdClub, xNombre);

setIdClub(club, xIdClub);

setNombre(club, xNombre);

if(localizarDato(listaClubes, club) == NULL){

adicionarFinal(listaClubes, club);

}

else{

cout << "El club " << getIdClub(club) << " ya existe" << endl;

fprintf(errores, "E100-0 | El club %d ya existe.\n", getIdClub(club));

}

}

destructor(club);

fclose(pf);

fclose(errores);

cout << "Se cargaron los clubes correctamente" << endl;

}

**void cargarJugadores(ListaJ &listaJugadores, ListaC &listaClubes, FILE \*pf, FILE \*errores)**{

Jugador jugador;

constructor(jugador);

Club club;

constructor(club);

bool seCargoUno = false;

int xIdJugador, xIdClub;

float xValor;

char xNombre[110];

char xPosicion[3];

PtrNodoListaC ptr;

PtrNodoListaJ ptrJ = primero(listaJugadores);

if ((errores = fopen("errores.txt", "a")) == NULL){

cout << "Error al abrir el archivo errores.txt" << endl;

Sleep(2000);

exit(0);

}

if ((pf = fopen("Jugadores.db", "r")) == NULL){

cout << "Error al abrir el archivo Jugadores.db" << endl;

Sleep(2000);

exit(0);

}

while(!feof(pf)){

fscanf(pf, "%d;%[A-Z a-z 0-9 .' ä-ü á-ú Á-Ú];%d;%[A-Z a-z 0-9 .' ä-ü á-ú];%f\n", &xIdJugador, xNombre, &xIdClub, xPosicion, &xValor);

setIdJugador(jugador, xIdJugador);

setNombre(jugador, xNombre);

setValor(jugador, xValor);

if(strcmpi(xPosicion,"ARQ") == 0) setPosicion(jugador, ARQ);

if(strcmpi(xPosicion,"DEF") == 0) setPosicion(jugador, DEF);

if(strcmpi(xPosicion,"MED") == 0) setPosicion(jugador, MED);

if(strcmpi(xPosicion,"DEL") == 0) setPosicion(jugador, DEL);

setIdClub(club, xIdClub);

ptr = localizarDato(listaClubes, club);

ptrJ = localizarDato(listaJugadores, jugador); // validar si jugador esta duplicado

if (ptrJ == NULL){

if(ptr!=NULL){

adicionarFinal(listaJugadores, jugador);

seCargoUno = true;

}

else{

cout << "Error: No se cargo el jugador " << getIdJugador(jugador) << " en el club "<< getIdClub(club) << ". Revisar errores.txt" << endl;

fprintf(errores, "E200-0 | No existe el club %d para agregar al jugador %d\n", getIdClub(club), getIdJugador(jugador));

}

}else{ // jugador no existe

cout << "Error: No se cargo el jugador " << getIdJugador(jugador) << " en el club "<< getIdClub(club) << ". Revisar errores.txt" << endl;

fprintf(errores, "E201-0 | El jugador %d se encuentra duplicado\n", getIdJugador(jugador));

}

}

destructor(club);

destructor(jugador);

fclose(pf);

fclose(errores);

if(seCargoUno == true) cout << "Se cargaron los jugadores correctamente." << endl;

if(seCargoUno == false) cout << "No se cargo ningun jugador." << endl;

}

**void cargarEquipos(ListaE &listaEquipos, FILE \*pf, FILE \*errores)**{

Equipo equipo;

constructor(equipo);

int xIdEquipo;

char xNombreEquipo[60];

char xNombreUsuario[60];

if ((errores = fopen("errores.txt", "a")) == NULL){

cout << "Error al abrir el archivo errores.txt" << endl;

Sleep(2000);

exit(0);

}

if ((pf = fopen("Equipos.db", "r")) == NULL){

cout << "Error al abrir el archivo Equipos.db" << endl;

Sleep(2000);

exit(0);

}

while(!feof(pf)){

fscanf(pf, "%d;%[A-Z a-z 0-9];%[A-Z a-z 0-9 .' ä-ü á-ú]\n", &xIdEquipo, xNombreEquipo, xNombreUsuario);

setIdEquipo(equipo, xIdEquipo);

setNombreEquipo(equipo, xNombreEquipo);

setNombreUsuario(equipo, xNombreUsuario);

if(localizarDato(listaEquipos, equipo) == NULL){

adicionarFinal(listaEquipos, equipo);

}

else{

cout << "El equipo " << getIdEquipo(equipo) << " ya existe" << endl;

fprintf(errores, "E300-0 | El equipo %d ya existe\n", getIdEquipo(equipo));

}

}

destructor(equipo);

fclose(pf);

fclose(errores);

cout << "Se cargaron los equipos correctamente" << endl;

}

**void cargarJugadoresPorEquipo(ListaE &listaEquipos, ListaJ &listaJugadores, FILE \*pf, FILE \*errores, Configuracion config)**{

Equipo equipo;

constructor(equipo);

Jugador jugador;

constructor(jugador);

bool equipoValido=0, jugadorValido=0;

int xIdEquipo;

int xIdJugador;

PtrNodoListaE ptrE;

PtrNodoListaJ ptrJ;

if ((errores = fopen("errores.txt", "a")) == NULL){

cout << "Error al abrir el archivo errores.txt" << endl;

Sleep(2000);

exit(0);

}

if ((pf = fopen("JugadoresPorEquipo.db", "r")) == NULL){

cout << "Error al abrir el archivo JugadoresPorEquipo.db" << endl;

Sleep(2000);

exit(0);

}

while(!feof(pf)){

fscanf(pf, "%d;%d\n", &xIdEquipo, &xIdJugador);

setIdEquipo(equipo, xIdEquipo);

setIdJugador(jugador, xIdJugador);

ptrE = localizarDato(listaEquipos, equipo);

ptrJ = localizarDato(listaJugadores, jugador);

if(ptrE && ptrJ != NULL){

obtenerDato(listaEquipos, equipo, ptrE);

obtenerDato(listaJugadores, jugador, ptrJ);

//Seteando monto y validando que no se pase del maximo

setMonto(equipo, (getMonto(equipo))+getValor(jugador));

equipoValido = validarValorEquipo(equipo, config, errores);

if(equipoValido){

if(getPosicion(jugador) == ARQ){

if(localizarDato(getListaARQ(equipo), jugador) == NULL){ // validar si jugador ya esta cargado en el equipo

adicionarFinal(getListaARQ(equipo), jugador);

jugadorValido = validarJugador(equipo, config, errores, ARQ);

}

else{

cout << "ERROR: El jugador " << getIdJugador(jugador) << " ya esta cargado en el equipo: " << getIdEquipo(equipo) << endl;

fprintf(errores, "E400-1 | El jugador %d ya esta cargado en el equipo %d\n", getIdJugador(jugador), getIdEquipo(equipo));

}

}

if(getPosicion(jugador) == DEF){

if(localizarDato(getListaDEF(equipo), jugador) == NULL){ // validar si jugador ya esta cargado en el equipo

adicionarFinal(getListaDEF(equipo), jugador);

jugadorValido = validarJugador(equipo, config, errores, DEF);

}

else{

cout << "ERROR: El jugador " << getIdJugador(jugador) << " ya esta cargado en el equipo: " << getIdEquipo(equipo) << endl;

fprintf(errores, "E400-2 | El jugador %d ya esta cargado en el equipo %d\n", getIdJugador(jugador), getIdEquipo(equipo));

}

}

if(getPosicion(jugador) == MED){

if(localizarDato(getListaMED(equipo), jugador) == NULL){ // validar si jugador ya esta cargado en el equipo

adicionarFinal(getListaMED(equipo), jugador);

jugadorValido = validarJugador(equipo, config, errores, MED);

}

else{

cout << "ERROR: El jugador " << getIdJugador(jugador) << " ya esta cargado en el equipo: " << getIdEquipo(equipo) << endl;

fprintf(errores, "E400-3 | El jugador %d ya esta cargado en el equipo %d\n", getIdJugador(jugador), getIdEquipo(equipo));

}

}

if(getPosicion(jugador) == DEL){

if(localizarDato(getListaDEL(equipo), jugador) == NULL){ // validar si jugador ya esta cargado en el equipo

adicionarFinal(getListaDEL(equipo), jugador);

jugadorValido = validarJugador(equipo, config, errores, DEL);

}

else{

cout << "ERROR: El jugador " << getIdJugador(jugador) << " ya esta cargado en el equipo: " << getIdEquipo(equipo) << endl;

fprintf(errores, "E400-4 | El jugador %d ya esta cargado en el equipo %d\n", getIdJugador(jugador), getIdEquipo(equipo));

}

}

if(jugadorValido) colocarDato(listaEquipos, equipo, ptrE);

}

if(!equipoValido || !jugadorValido) eliminarDato(listaEquipos, equipo);

}

else { //errores.txt

if(ptrE == NULL){

printf("ERROR: El equipo con numero %d no existe. Revisar errores.txt\n", getIdEquipo(equipo));

fprintf(errores, "E403-0 | El equipo con numero %d no existe\n", getIdEquipo(equipo));

}

if(ptrJ == NULL){

printf("ERROR: El jugador con numero %d no existe. Revisar errores.txt\n", getIdJugador(jugador));

fprintf(errores, "E404-0 | El jugador con numero %d no existe. Revisar Jugadores.db\n", getIdJugador(jugador));

}

}

}

destructor(jugador);

destructor(equipo);

fclose(errores);

fclose(pf);

cout << "Se cargaron los jugadores por equipo correctamente" << endl;

}

**void cargarPuntosPorJugador(ListaE &listaEquipos, ListaJ &listaJugadores, Cola &colaPuntosFecha, FILE \*pf, FILE \*errores)**{

PuntosFecha puntosFecha;

constructor(puntosFecha);

int flagFecha = 0;

int i=0;

Equipo equipo;

constructor(equipo);

Jugador jugador;

constructor(jugador);

PuntosEquipo puntoEquipo;

PtrNodoListaE ptrLista;

ptrNodoCola ptrUltimo;

ptrNodoCola ptrCola;

PtrNodoListaJ ptrJugador = NULL;

string aux, aux2;

int fechaAux=1; //ACA SE DEBE PONER LA FECHA EN LA CUAL INICIA EL TORNEO

int xIdJugador, xFecha, xPuntos;

if ((pf = fopen("PuntosPorJugador.db", "r")) == NULL){

cout << "Error al abrir el archivo Equipos.db" << endl;

Sleep(2000);

exit(0);

}

while(!feof(pf)){ // Cargando cola para trabajar directamente de ella

fscanf(pf,"%d;%d;%d", &xIdJugador, &xFecha, &xPuntos);

setIdJugador(puntosFecha, xIdJugador);

setFecha(puntosFecha, xFecha);

setPuntos(puntosFecha, xPuntos);

encolar(colaPuntosFecha, puntosFecha);

}

ptrUltimo = ultimo(colaPuntosFecha);

puntosFecha = getDato(ptrUltimo);

int ultimaFecha = getFecha(puntosFecha);

PuntosEquipo puntosEquipo[(longitud(listaEquipos)\*ultimaFecha)]; //CREANDO ARRAY DE ESTRUCTURA

while(fechaAux<=ultimaFecha){ //WHILE GENERAL PARA IR RECORRIENDO TODAS LAS FECHAS

puntoEquipo.fecha = fechaAux; //ASIGNO LA FECHA DE LA CUAL SE CONSULTAN LOS PUNTAJES DE LOS EQUIPOS

/////////////SE CREA EL ARCHIVO DE LA FECHA//////////////////

/\*aux = static\_cast<std::ostringstream\*>(&(std::ostringstream() << fechaAux))->str();

aux2 = "PuntosFecha"+aux+".txt";

const char \* nombreArchivo = aux2.c\_str();

if ((pf = fopen(nombreArchivo,"a")) == NULL){

cout << "Error al abrir el archivo Fecha" << fechaAux << ".db" << endl;

Sleep(2000);

exit(0);

}\*/

////////////////////////////////////////////////////////////

ptrLista = primero(listaEquipos);

while(ptrLista!=NULL){

puntoEquipo.puntos = 0;

obtenerDato(listaEquipos, equipo, ptrLista); //TRAIGO AL EQUIPO Y LO METO EN LA INSTANCIA TDA EQUIPO

puntoEquipo.idEquipo = getIdEquipo(equipo); //ASIGNO EL ID DE EQUIPO DEL QUE ESTOY CONSULTANDO EL PUNTAJE

ptrCola = primero(colaPuntosFecha);

flagFecha=0;

while(ptrCola!=NULL){

puntosFecha = getDato(ptrCola); //METO EN LA INSTANCIA PUNTOSFECHA EL PRIMER ELEMENTO DE LA COLA

if(getFecha(puntosFecha) != fechaAux) flagFecha=1; //SI LA FECHA CAMBIO NO LA PROCESA

setIdJugador(jugador, getIdJugador(puntosFecha)); //SETEO EL ID PARA BUSCARLO EN LA LISTA

ptrJugador = localizarDato(listaJugadores, jugador); //LO BUSCO EN LA LISTA Y DEVUELVO EL PUNTERO

if(ptrJugador!=NULL && flagFecha==0){

obtenerDato(listaJugadores, jugador, ptrJugador); //OBTENGO EL JUGADOR Y LO GUARDO EN LA INSTANCIA TDA JUGADOR

ptrJugador=NULL; //SE SETEA EN NULL DE NUEVO PARA BUSCARLO EN LAS LISTAS DEL EQUIPO

if(getPosicion(jugador)==ARQ) ptrJugador=localizarDato(getListaARQ(equipo), jugador);//ES ARQUERO DE ESTE EQUIPO?

if(getPosicion(jugador)==DEF) ptrJugador=localizarDato(getListaDEF(equipo), jugador);//ES DEFENSOR DE ESTE EQUIPO?

if(getPosicion(jugador)==MED) ptrJugador=localizarDato(getListaMED(equipo), jugador);//ES MEDIOCAMPISTA DE ESTE EQUIPO?

if(getPosicion(jugador)==DEL) ptrJugador=localizarDato(getListaDEL(equipo), jugador);//ES DELANTERO DE ESTE EQUIPO?

if(ptrJugador!=NULL && flagFecha==0){

/\*cout << "Fecha: " << fechaAux << " Jugador: ";

cout << getIdJugador(jugador);

cout << " Equipo: ";

cout << puntoEquipo.idEquipo << endl;\*/

puntoEquipo.puntos += getPuntos(puntosFecha); //SUMO EL PUNTAJE DEL JUGADOR SELECCIONADO EN EL EQUIPO

}

}

ptrCola = siguiente(ptrCola); //SE VE EL SIGUIENTE ELEMENTO DE PUNTOSFECHA

flagFecha=0;

} //FIN WHILE PUNTOSFECHA

//IMPRIMIENDO EL ARCHIVO SIN METODO QUE BUGUEE TODO

//imprimirOrdenado(equipo, fechaAux, puntoEquipo, pf);

puntosEquipo[i].idEquipo = puntoEquipo.idEquipo;

puntosEquipo[i].nomEquipo = getNombreEquipo(equipo);

puntosEquipo[i].nomUsuario = getNombreUsuario(equipo);

puntosEquipo[i].fecha = puntoEquipo.fecha;

puntosEquipo[i].puntos = puntoEquipo.puntos;

i++;

ptrLista = siguiente(listaEquipos, ptrLista); //SE ANALIZAN LOS PUNTOS DEL SIGUIENTE EQUIPO

} //FIN WHILE EQUIPOS

fechaAux++;

fclose(pf);

} //FIN WHILE FECHA

imprimirOrdenado(puntosEquipo, ultimaFecha, listaEquipos, listaJugadores, colaPuntosFecha); //LLAMADA A PROCEDIMIENTO PARA ORDENAR E IMPRIMIR LO CAPTURADO

cout << "Se cargaron los puntos por jugador correctamente" << endl;

}

**void imprimirOrdenado(PuntosEquipo puntosEquipo[], int ultimaFecha, ListaE &listaEquipos, ListaJ &listaJugadores, Cola &colaPuntosFecha)**{ //SE IMPRIME Y ORDENA EL PUNTAJE DE CADA EQUIPO EN LA FECHA CORRESPONDIENTE

int i;

int j;

int cursor=0;

FILE \*pf;

FILE \*pf2;

FILE \*pf3;

PuntosEquipo puntosEquipoTotal[longitud(listaEquipos)];

PuntosEquipo auxiliar;

int fechaAux=1;

string aux,aux2;

/\*while(fechaAux<=ultimaFecha){

aux = static\_cast<std::ostringstream\*>(&(std::ostringstream() << fechaAux))->str();

aux2 = "PuntosFecha"+aux+".txt";

const char \* nombreArchivo = aux2.c\_str();

if ((pf = fopen(nombreArchivo,"a")) == NULL){

cout << "Error al abrir el archivo Fecha" << fechaAux << ".txt" << endl;

Sleep(2000);

exit(0);

}\*/

/////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////CARGAS Y ORDENAMIENTOS////////////////////////////////////

//SE ORDENA LA ESTRUCTURA DE ARRAY PARA LOS PUNTOS DE LA FECHA POR EQUIPO (YA CARGADA)

for(i=0 ; i<(longitud(listaEquipos)\*ultimaFecha) ; i++){

for(j=0 ; j<(longitud(listaEquipos)\*ultimaFecha)-1 ; j++){

if(puntosEquipo[j].puntos < puntosEquipo[j+1].puntos && puntosEquipo[j].fecha == puntosEquipo[j+1].fecha){

auxiliar = puntosEquipo[j];

puntosEquipo[j] = puntosEquipo[j+1];

puntosEquipo[j+1] = auxiliar;

}

}

}

//SE CARGA LA ESTRUCTURA DE ARRAY PARA LOS PUNTOS POR EQUIPO EN TODO EL TORNEO (LA FECHA SE IGNORA, =0)

for(i=0 ; i<(longitud(listaEquipos)) ; i++){

for(j=0 ; j<(longitud(listaEquipos)\*ultimaFecha) ; j++){

if(j==i){

puntosEquipoTotal[i].idEquipo = puntosEquipo[j].idEquipo;

puntosEquipoTotal[i].puntos = puntosEquipo[j].puntos;

puntosEquipoTotal[i].nomEquipo = puntosEquipo[j].nomEquipo;

puntosEquipoTotal[i].nomUsuario = puntosEquipo[j].nomUsuario;

puntosEquipoTotal[i].fecha = 0;

}

if(j>i && puntosEquipo[j].idEquipo == puntosEquipoTotal[i].idEquipo){

puntosEquipoTotal[i].puntos+=puntosEquipo[j].puntos; //Se suma el puntaje del resto de fechas

}

}

}

//SE ORDENA LA ESTRUCTURA DE ARRAY PARA LOS PUNTOS POR EQUIPO EN TODO EL TORNEO:

for(i=0 ; i<(longitud(listaEquipos)) ; i++){

for(j=0 ; j<(longitud(listaEquipos))-1 ; j++){

if(puntosEquipoTotal[j].puntos < puntosEquipoTotal[j+1].puntos && puntosEquipoTotal[j].fecha == puntosEquipoTotal[j+1].fecha){

auxiliar = puntosEquipoTotal[j];

puntosEquipoTotal[j] = puntosEquipoTotal[j+1];

puntosEquipoTotal[j+1] = auxiliar;

}

}

}

//SE AGRUPAN TODOS LOS JUGADORES CON SUS PUNTOS TOTALES (FECHA = 0 YA QUE NO IMPORTA)

Cola colaOrdenada;

crearCola(colaOrdenada);

PuntosFecha pfAux;

PuntosFecha pfMax;

PuntosFecha pfCursor;

ptrNodoCola cursorCola = primero(colaPuntosFecha);

ptrNodoCola auxCola = primero(colaPuntosFecha);

int encontrado = 0;

while(cursorCola!=NULL){

pfCursor=getDato(cursorCola);

while(auxCola !=NULL && encontrado==0){

pfAux = getDato(auxCola);

if(getIdJugador(pfCursor)==getIdJugador(pfAux) && cursorCola!=auxCola){

cursorCola->ptrdato.puntos += getPuntos(pfAux);

eliminarNodo(colaPuntosFecha, auxCola);

encontrado = 1;

}

auxCola=siguiente(auxCola);

}

encontrado=0;

auxCola=primero(colaPuntosFecha);

cursorCola=siguiente(cursorCola);

}

ptrNodoCola ptrPruebita = primero(colaPuntosFecha);

/\*while(ptrPruebita!=NULL){

pfAux = getDato(ptrPruebita);

cout << "Jugador: " << getIdJugador(pfAux) << " / Puntos: " << getPuntos(pfAux) << endl;;

ptrPruebita = siguiente(ptrPruebita);

}\*/

//SE ORDENA LA COLA DE JUGADORES POR PUNTAJE

setPuntos(pfMax, -100);

constructor(pfAux);

int flagcontrol;

ptrNodoCola ptrControl;

ptrNodoCola ptrCola = primero(colaPuntosFecha);

ptrNodoCola ptrMax = primero(colaPuntosFecha);

while(!colaVacia(colaPuntosFecha)){

while(ptrCola!=NULL){

pfAux = getDato(ptrCola);

if(getPuntos(pfAux)>getPuntos(pfMax)){

pfMax = pfAux;

ptrMax = ptrCola;

/\*while(ptrCola!=ptrMax){

ptrMax=siguiente(ptrMax);

}\*/

}

ptrCola=siguiente(ptrCola);

}

encolar(colaOrdenada, pfMax);

eliminarNodo(colaPuntosFecha, ptrMax);

ptrCola = primero(colaPuntosFecha);

ptrMax = primero(colaPuntosFecha);

setPuntos(pfMax, -100);

}

/\*ptrNodoCola ptrPruebita = primero(colaOrdenada);

while(ptrPruebita!=NULL){

pfAux = getDato(ptrPruebita);

cout << "Jugador: " << getIdJugador(pfAux) << " / Puntos: " << getPuntos(pfAux) << endl;;

ptrPruebita = siguiente(ptrPruebita);

}\*/

/////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////IMPRESIONES//////////////////////////////////////////

//SE IMPRIME EL ARCHIVO DE EQUIPOS EN TODO EL TORNEO ORDENADO POR PUNTOS

if ((pf2 = fopen("puntajesTotalesPorEquipo.txt","w")) == NULL){

cout << "Error al abrir el archivo puntajesTotalesPorEquipo.txt" << endl;

Sleep(2000);

exit(0);

}

for(i=0 ; i<longitud(listaEquipos) ; i++){

fprintf(pf2,"%d;%d;%s;%s;%d\n", i+1, puntosEquipoTotal[i].idEquipo, puntosEquipoTotal[i].nomEquipo.c\_str(), puntosEquipoTotal[i].nomUsuario.c\_str(), puntosEquipoTotal[i].puntos);

}

fclose(pf2);

//SE IMPRIMEN LOS ARCHIVOS DE FECHA PREVIAMENTE ORDENADOS POR PUNTOS

while (fechaAux <= ultimaFecha){

aux = static\_cast<std::ostringstream\*>(&(std::ostringstream() << fechaAux))->str();

aux2 = "puntajesFecha"+aux+".txt";

const char \* nombreArchivo = aux2.c\_str();

if ((pf = fopen(nombreArchivo,"a")) == NULL){

cout << "Error al abrir el archivo puntajesFecha" << fechaAux << ".txt" << endl;

Sleep(2000);

exit(0);

}

for(i=0 ; i<longitud(listaEquipos) ; i++){

fprintf(pf, "%d;%d;%s;%s;%d\n", i+1, puntosEquipo[cursor].idEquipo, puntosEquipo[cursor].nomEquipo.c\_str(), puntosEquipo[cursor].nomUsuario.c\_str(), puntosEquipo[cursor].puntos);

cursor++;

}

fechaAux++;

fclose(pf);

}

//SE IMPRIME EL RANKING DE JUGADORES ORDENADO POR PUNTOS

if ((pf3 = fopen("puntajesPorJugador.txt","w")) == NULL){

cout << "Error al abrir el archivo puntajesTotalesPorEquipo.txt" << endl;

Sleep(2000);

exit(0);

}

ptrNodoCola ptrCursor = primero(colaOrdenada);

PtrNodoListaJ control;

Jugador jugador;

constructor(jugador);

i=1;

while(ptrCursor!=NULL){

pfAux = getDato(ptrCursor);

setIdJugador(jugador, getIdJugador(pfAux));

control = localizarDato(listaJugadores, jugador);

if(control!=NULL){

fprintf(pf3, "%d;%d;%d\n", i, getIdJugador(pfAux), getPuntos(pfAux));

i++;

}

ptrCursor = siguiente(ptrCursor);

}

fclose(pf3);

eliminarCola(colaOrdenada);

//Destruyendo TDAs, listas y colas:

destructor(pfMax);

destructor(pfAux);

destructor(pfCursor);

destructor(jugador);

eliminarCola(colaOrdenada);

}

// COSAS AUXILIARES PARA MOSTRAR ------------------------------------------------------------------------------

**bool traerClubes(ListaC &listaClubes)**{

system("cls");

cout << "Clubes\n" << endl;

Club aux;

constructor(aux);

PtrNodoListaC ptr = primero(listaClubes);

while(ptr!=NULL){

obtenerDato(listaClubes, aux, ptr);

cout << "ID Club: " << getIdClub(aux) << " | Nombre: " << getNombre(aux) << endl;

ptr = siguiente(listaClubes, ptr);

}

destructor(aux);

return true;

}

**bool traerJugadores(ListaJ &listaJugadores)**{

Jugador aux;

constructor(aux);

PtrNodoListaJ ptrJ = primero(listaJugadores);

while(ptrJ!=NULL){

obtenerDato(listaJugadores, aux, ptrJ);

cout << "ID jugador: " << getIdJugador(aux) << " | Nombre: " << getNombre(aux) << " | Posicion: " << getPosicion(aux) << endl;

ptrJ = siguiente(listaJugadores, ptrJ);

}

destructor(aux);

return true;

}

**bool traerEquipos(ListaE &listaEquipos)**{

system("cls");

cout << "Equipos\n" << endl;

Equipo aux;

constructor(aux);

PtrNodoListaE ptr = primero(listaEquipos);

while(ptr!=NULL){

obtenerDato(listaEquipos, aux, ptr);

cout << "ID equipo: " << getIdEquipo(aux) << " | Nombre: " << getNombreEquipo(aux) << " | Usuario: " << getNombreUsuario(aux) << endl;

ptr = siguiente(listaEquipos, ptr);

}

destructor(aux);

return true;

}

**bool traerJugadorePorEquipo(ListaE &listaEquipos)**{

system("cls");

cout << "Jugadores por equipos\n" << endl;

Equipo auxE;

constructor(auxE);

PtrNodoListaE ptrE = primero(listaEquipos);

while(ptrE!=NULL){

obtenerDato(listaEquipos, auxE, ptrE);

cout << "ID equipo: " << getIdEquipo(auxE) << " | Nombre: " << getNombreEquipo(auxE) << " | Usuario: " << getNombreUsuario(auxE) <<"\n"<< endl;

cout << "Arqueros: " << endl;

traerJugadores(getListaARQ(auxE));

cout << "Defensores: " << endl;

traerJugadores(getListaDEF(auxE));

cout << "Mediocampistas: " << endl;

traerJugadores(getListaMED(auxE));

cout << "Delanteros: " << endl;

traerJugadores(getListaDEL(auxE));

cout << endl;

/\*

PtrNodoListaJ ptrJ = primero(getListaARQ(auxE));

while(ptrJ!=NULL){

obtenerDato(getListaARQ(auxE),auxJ, ptrJ);

cout << "ID jugador: " << getIdJugador(auxJ) << " | Nombre: " << getNombre(auxJ) << " | Posicion: " << getPosicion(auxJ) << endl;

ptrJ = siguiente(getListaARQ(auxE), ptrJ);

}

\*/

ptrE = siguiente(listaEquipos, ptrE);

}

//Destruyendo TDAs

destructor(auxE);

return true;

}

**bool validarJugador(Equipo &equipo, Configuracion config, FILE \*errores, Posicion posicion)**{

bool valido = 0;

PtrNodoListaJ ptrJ;

int numero = 0;

Jugador jugador;

constructor(jugador);

if(posicion==ARQ){

if(longitud(getListaARQ(equipo))>config.maxArq){

printf("ERROR: Se descalifica equipo %d por superar el maximo de arqueros. Revisar errores.txt\n", getIdEquipo(equipo));

fprintf(errores, "\nE401-1 | Se descalifica equipo %d por tener %d arqueros. (MAXIMO: %d)\n", getIdEquipo(equipo), longitud(getListaARQ(equipo)), config.maxArq);

ptrJ = primero(getListaARQ(equipo));

while (ptrJ != NULL){

numero ++;

obtenerDato(getListaARQ(equipo), jugador, ptrJ);

fprintf(errores, "\t| %d -> ID: %d | Nombre: %s | Posicion: %d\n", numero, getIdJugador(jugador), getNombre(jugador).c\_str(), getPosicion(jugador));

ptrJ = siguiente(getListaARQ(equipo), ptrJ);

}

fprintf(errores,"\n");

}

else valido=1;

}

if(posicion==DEF){

if(longitud(getListaDEF(equipo))>config.maxDef){

printf("ERROR: Se descalifica equipo %d por superar el maximo de defensores. Revisar errores.txt\n", getIdEquipo(equipo));

fprintf(errores, "\nE401-2 | Se descalifica equipo %d por tener %d defensores. (MAXIMO: %d)\n", getIdEquipo(equipo), longitud(getListaDEF(equipo)), config.maxDef);

ptrJ = primero(getListaDEF(equipo));

while (ptrJ != NULL){

numero ++;

obtenerDato(getListaDEF(equipo), jugador, ptrJ);

fprintf(errores, "\t| %d -> ID: %d | Nombre: %s | Posicion: %d\n", numero, getIdJugador(jugador), getNombre(jugador).c\_str(), getPosicion(jugador));

ptrJ = siguiente(getListaDEF(equipo), ptrJ);

}

fprintf(errores,"\n");

}

else valido=1;

}

if(posicion==MED){

if(longitud(getListaMED(equipo))>config.maxMed){

printf("ERROR: Se descalifica equipo %d por superar el maximo de mediocampistas. Revisar errores.txt\n", getIdEquipo(equipo));

fprintf(errores, "\nE402-3 | Se descalifica equipo %d por tener %d mediocampistas. (MAXIMO: %d)\n", getIdEquipo(equipo), longitud(getListaMED(equipo)), config.maxMed);

ptrJ = primero(getListaMED(equipo));

while (ptrJ != NULL){

numero ++;

obtenerDato(getListaMED(equipo), jugador, ptrJ);

fprintf(errores, "\t| %d -> ID: %d | Nombre: %s | Posicion: %d\n", numero, getIdJugador(jugador), getNombre(jugador).c\_str(), getPosicion(jugador));

ptrJ = siguiente(getListaMED(equipo), ptrJ);

}

fprintf(errores,"\n");

}

else valido=1;

}

if(posicion==DEL){

if(longitud(getListaDEL(equipo))>config.maxDel){

printf("ERROR: Se descalifica equipo %d por superar el maximo de delanteros. Revisar errores.txt\n", getIdEquipo(equipo));

fprintf(errores, "\nE402-4 | Se descalifica equipo %d por tener %d delanteros. (MAXIMO: %d)\n", getIdEquipo(equipo), longitud(getListaDEL(equipo)), config.maxDel);

ptrJ = primero(getListaDEL(equipo));

while (ptrJ != NULL){

numero ++;

obtenerDato(getListaDEL(equipo), jugador, ptrJ);

fprintf(errores, "\t| %d -> ID: %d | Nombre: %s | Posicion: %d\n", numero, getIdJugador(jugador), getNombre(jugador).c\_str(), getPosicion(jugador));

ptrJ = siguiente(getListaDEL(equipo), ptrJ);

}

fprintf(errores,"\n");

}

else valido=1;

}

return valido;

}

**bool validarValorEquipo(Equipo &equipo, Configuracion config, FILE \*errores)**{

bool valido = 0;

if(getMonto(equipo)>config.ValorMaximoXEquipo){

printf("ERROR: Se descalifica equipo %d por superar el valor maximo permitido. Revisar errores.txt\n", getIdEquipo(equipo));

fprintf(errores, "E405-0 | Se descalifica equipo %d por tener un valor de %.2f (MAXIMO: %.2f)\n", getIdEquipo(equipo), getMonto(equipo), config.ValorMaximoXEquipo);

}

else valido = 1;

return valido;

}

1;Aldosivi

2;Arsenal

3;Atl. de Tucumán

4;Banfield

5;Belgrano

6;Boca

7;Colón

8;Def. y Justicia

9;Estudiantes

10;Gimnacia L.P.

11;Godoy Cruz

12;Huracán

13;Independiente

14;Lanús

15;Newell's

16;Olimpo

17;Patronato

18;Quilmes

19;Racing

20;Rafaela

21;River

22;Rosario Central

23;San Lorenzo

24;San Martin SJ

25;Sarmiento

26;Talleres

27;Temperley

28;Unión

29;Velez

1;Campodónico Pablo;1;Arq;1500000

2;Gutierrez Matias;1;Arq;2500000

24;Pellegrino Fernando;2;Arq;2500000

25;Santillo Pablo;2;Arq;2500000

44;Ayala Josue;3;Arq;1500000

45;Lucchetti Cristian;3;Arq;2500000

66;Arboleda Iván;4;Arq;2500000

67;Navarro Hilario;4;Arq;3500000

87;Acosta Lucas;5;Arq;3500000

105;Rossi Agustín;6;Arq;4500000

124;Broun Jorge;7;Arq;2500000

144;Arias Gabriel;8;Arq;5500000

164;Andújar Mariano;9;Arq;4500000

165;Sappa Daniel;9;Arq;4500000

185;Martín Arias Alexis;10;Arq;3500000

207;Rey Rodrigo;11;Arq;2500000

232;Díaz Marcos;12;Arq;1500000

233;Marinelli Gonzalo;12;Arq;1500000

252;Campaña Martín;13;Arq;2500000

3;Alegre Alan Jesús;1;Def;1500000

4;Arias Ramiro;1;Def;2500000

5;Canever Franco;1;Def;2500000

6;Galván Jonatan;1;Def;3500000

7;Ledesma Damián;1;Def;2500000

8;Quílez Ismael;1;Def;2500000

9;Sosa Otermin Leandro;1;Def;3500000

26;Bottinelli Jonathan;2;Def;2500000

27;Corvalán Claudio;2;Def;2500000

28;Curado Marcos;2;Def;1500000

29;Marín Leandro;2;Def;2500000

46;Bianchi Bruno;3;Def;2500000

47;Canuto Ignacio;3;Def;2500000

48;Di Plácido Leonel;3;Def;1500000

49;Evangelista Fernando;3;Def;2500000

50;Meza Brítez Enrique;3;Def;2500000

51;Rosales Mauricio;3;Def;1500000

68;Bettini Gonzalo;4;Def;3500000

69;Civelli Renato;4;Def;2500000

70;Matheu Carlos;4;Def;2500000

71;Rodríguez Jorge A.;4;Def;2500000

72;Soto Alexis;4;Def;2500000

73;Sporle Adrián;4;Def;2500000

88;Aveldaño Lucas;5;Def;2500000

89;Espíndola López Estebán;5;Def;2500000

90;Lema Cristian;5;Def;2500000

91;Luna Sebastián;5;Def;3500000

92;Quiroga Juan;5;Def;1500000

93;Romero Cristian;5;Def;2500000

94;Saravia Renzo;5;Def;2500000

106;Fabra Frank;6;Def;2500000

107;Insaurralde Juan M.;6;Def;2500000

108;Peruzzi Gino;6;Def;3500000

109;Silva Jonathan;6;Def;2500000

110;Vergini Santiago;6;Def;3500000

125;Ceballos Lucas;7;Def;2500000

126;Conti Germán;7;Def;2500000

127;Garnier Yamil;7;Def;2500000

128;Olivera Emanuel;7;Def;2500000

129;Ortiz Guillermo Luis;7;Def;2500000

130;Rodríguez Clemente;7;Def;2500000

145;Alvarado Pablo;8;Def;4500000

146;Barboza Ullúa Alexander;8;Def;2500000

147;Barrios Rafael;8;Def;2500000

148;Cardona Tomás;8;Def;2500000

149;Delgado Rafael;8;Def;3500000

150;Gómez Elías;8;Def;2500000

151;Silva Hugo;8;Def;2500000

166;Aguirregaray Matías;9;Def;3500000

167;Desábato Leandro;9;Def;2500000

168;Diarte Lucas;9;Def;1500000

169;Dubarbier Sebastián;9;Def;2500000

170;Foyth Juan;9;Def;1500000

171;Sánchez Facundo;9;Def;2500000

172;Schunke Jonatan;9;Def;3500000

185;Ceceri Kevin;10;Def;2500000

186;Gorga Sebastián;10;Def;2500000

187;Guanini Manuel;10;Def;4500000

188;Licht Lucas;10;Def;4500000

189;Oreja Facundo;10;Def;1500000

190;Romero Mauricio;10;Def;1500000

208;Abecasis Luciano;11;Def;1500000

209;Benítez Marcelo;11;Def;3500000

210;Bonacorso Ezequiel;11;Def;1500000

211;Cobos Facundo;11;Def;1500000

212;Galeano Leonel;11;Def;2500000

213;Olivares Sebastián;11;Def;2500000

214;Ortiz Danilo;11;Def;1500000

215;Pizarro Luciano;11;Def;3500000

216;Viera Diego;11;Def;3500000

234;Araujo Carlos;12;Def;3500000

235;Garré Ezequiel;12;Def;4500000

236;Mancinelli Federico;12;Def;2500000

237;Nervo Martín;12;Def;2500000

238;Risso Mario Pablo;12;Def;2500000

239;Romat Nicolás;12;Def;2500000

240;Villalba Lucas;12;Def;4500000

253;Bustos Fabricio;13;Def;3500000

254;Figal Nicolás;13;Def;4500000

255;Franco Alan;13;Def;4500000

256;Tagliafico Nicolás;13;Def;2500000

16;Acosta Joel;1;Del;2500000

17;Bandiera Neri;1;Del;1500000

18;Benedetti Jonatan;1;Del;2500000

19;Leonardo Francisco;1;Del;1500000

20;Lugüercio Pablo;1;Del;2500000

21;Medina Antonio;1;Del;2500000

22;Miracco Nicolás;1;Del;2500000

23;Penco Sebastián;1;Del;3500000

40;Boghossian Joaquín;2;Del;2500000

41;Fragapane Franco;2;Del;1500000

42;Rodríguez Julio;2;Del;1500000

43;Sánchez Sotelo Juan;2;Del;2500000

63;Méndez José A.;3;Del;2500000

64;Menéndez Cristian;3;Del;2500000

65;Rodríguez Luis;3;Del;3500000

65;Zampedri Fernando;3;Del;2500000

84;Asenjo Mauricio;4;Del;2500000

85;Cvitanich Dario;4;Del;4500000

86;Fontana Agustín;4;Del;2500000

101;Bieler Claudio;5;Del;2500000

102;Márquez Fernando;5;Del;3500000

103;Melano Lucas;5;Del;2500000

104;Suárez Matías;5;Del;3500000

119;Benedetto Darío;6;Del;2500000

120;Benitez Oscar;6;Del;4500000

121;Bou Walter;6;Del;2500000

122;Pavón Cristian;6;Del;2500000

123;Solís Nazareno;6;Del;4500000

138;Blanco Ismael;7;Del;2500000

139;Lagos Diego;7;Del;2500000

140;Leguizamón Nicolás;7;Del;2500000

140;Pereyra Facundo;7;Del;2500000

142;Silva Nicolás;7;Del;2500000

143;Vera Diego;7;Del;2500000

160;Bouzat Agustín;8;Del;5500000

161;Kaprof Juan Cruz;8;Del;5500000

162;Ríos Andrés;8;Del;3500000

163;Stefanelli Nicolás;8;Del;2500000

180;tero Juan Ferney;9;Del;2500000

181;Perotti Juan;9;Del;4500000

182;Toledo Javier;9;Del;2500000

183;Vargas Jeisson;9;Del;2500000

184;Viatri Lucas;9;Del;4500000

202;Ibáñez Nicolás;10;Del;2500000

203;Matos Mauro;10;Del;4500000

204;Mazzola Nicolás;10;Del;2500000

205;Niell Franco;10;Del;3500000

206;Ramirez Erick Iván;10;Del;2500000

227;Correa Javier;11;Del;3500000

228;García Santiago;11;Del;2500000

229;Garro Juan;11;Del;3500000

230;Sigales Maximiliano;11;Del;1500000

231;Silva Facundo;11;Del;3500000

247;Briasco Norberto;12;Del;2500000

248;Chacana Lucas;12;Del;1500000

249;Mendoza Diego;12;Del;2500000

250;Molina Tomás;12;Del;1500000

251;Pussetto Ignacio;12;Del;2500000

263;Albertengo Lucas;13;Del;2500000

264;Benítez Martín;13;Del;3500000

265;Gigliotti Emanuel;13;Del;2500000

266;Togni Gastón;13;Del;4500000

10;Brum Roberto;1;Med;2500000

11;Castro Alexis N.;1;Med;2500000

12;Díaz Ángel Gastón;1;Med;2500000

13;Llama Cristian;1;Med;2500000

14;Navarro Sebastián;1;Med;3500000

15;Yeri Nahuel;1;Med;1500000

30;Bazán Gonzalo;2;Med;1500000

31;Bellocq Franco;2;Med;2500000

32;Brunetta Juan;2;Med;2500000

33;Milo Federico;2;Med;2500000

34;Papa Gonzalo;2;Med;1500000

35;Pérez Renso;2;Med;2500000

36;Rolon Leonardo;2;Med;2500000

37;Sanabria Gabriel;2;Med;1500000

38;Velázquez Sergio;2;Med;2500000

39;Wilchez Lucas;2;Med;1500000

51;Acosta Guillermo;3;Med;2500000

52;Aliendro Rodrigo;3;Med;1500000

53;Alvarez Favio;3;Med;2500000

54;Barbona David;3;Med;1500000

55;Cuello Tomás;3;Med;2500000

56;González Leandro;3;Med;2500000

57;Leyes Nery;3;Med;1500000

58;Mendoza Javier;3;Med;2500000

60;Ontivero Gonzalo;3;Med;2500000

61;Palomino Jairo;3;Med;2500000

62;Quiroga Franco;3;Med;4500000

74;Bertolo Nicolás;4;Med;4500000

75;Cecchini Emanuel;4;Med;2500000

76;Cobo Juan;4;Med;2500000

77;Colela Franco;4;Med;4500000

78;Gómez Luciano;4;Med;2500000

79;Lucero Martín;4;Med;2500000

80;Remedi Eric;4;Med;4500000

81;Rodríguez Thomas;4;Med;2500000

82;Sarmiento Brian;4;Med;2500000

83;Sperduti Mauricio;4;Med;2500000

95;Álvarez Suárez Gastón;5;Med;2500000

96;Aquino Claudio;5;Med;2500000

97;Barbieri Mariano;5;Med;2500000

98;Farré Guillermo;5;Med;3500000

99;Lértora Federico;5;Med;2500000

100;Velázquez Jorge;5;Med;2500000

111;Barrios Wilmar;6;Med;2500000

112;Bentancur Rodrigo;6;Med;2500000

113;Centurión Ricardo;6;Med;4500000

114;Gago Fernando;6;Med;2500000

115;Jara Leonardo;6;Med;4500000

116;Maroni Gonzalo;6;Med;2500000

117;Pérez Pablo;6;Med;2500000

118;Pérez Sebastián;6;Med;4500000

119;Zuqui Fernando;6;Med;2500000

131;Bastía Adrián;7;Med;2500000

132;Bauza Juan;7;Med;2500000

133;Bernardi Christian;7;Med;2500000

134;Ledesma Pablo;7;Med;2500000

135;Oviedo Fidencio;7;Med;2500000

136;Poblete Gerónimo;7;Med;2500000

137;Torres Iván Arturo;7;Med;2500000

152;Bareiro Mariano;8;Med;1500000

153;Castellani Gonzalo;8;Med;2500000

154;Elizari Fernando;8;Med;2500000

155;Fredes Hernán;8;Med;3500000

156;Gutiérrez Jonás;8;Med;2500000

157;Miranda Leonel;8;Med;2500000

158;Pochettino Tomás;8;Med;3500000

159;Rivero José Ignacio;8;Med;2500000

173;Ascacibar Santiago;9;Med;2500000

174;Braña Rodrigo;9;Med;2500000

175;Cascini Juan Bautista;9;Med;2500000

176;Cejas Juan Bautista;9;Med;3500000

177;Damonte Israel;9;Med;2500000

178;Gómez Iván;9;Med;2500000

179;Umeres Elías;9;Med;2500000

191;Aleman Brahian;10;Med;2500000

192;Bottinelli Darío;10;Med;2500000

193;Carrera Ramiro;10;Med;4500000

194;Faravelli Lorenzo;10;Med;2500000

195;Imperiale Daniel;10;Med;4500000

196;Lobos Lucas;10;Med;2500000

197;Noble Matías;10;Med;1500000

198;Perdomo Luciano;10;Med;2500000

199;Rinaudo Fabián;10;Med;1500000

200;Romero Sebastián;10;Med;2500000

217;Angileri Fabrizio;11;Med;3500000

218;Carabajal Gabriel;11;Med;1500000

219;Correa Maximiliano;11;Med;3500000

220;Fernández Guillermo;11;Med;3500000

221;Giménez Gastón;11;Med;4500000

222;González Angel;11;Med;4500000

223;Henríquez Fabián;11;Med;2500000

224;Sánchez Nicolás;11;Med;2500000

225;Serrano Walter;11;Med;1500000

226;Verdugo Agustín;11;Med;1500000

241;Angulo Julio;12;Med;2500000

242;Bogado Mauro;12;Med;4500000

243;Compagnucci Lucio;12;Med;2500000

244;Cuevas Lucas;12;Med;2500000

245;Fritzler Matías;12;Med;3500000

246;González Mariano;12;Med;3500000

257;Barco Ezequiel;13;Med;3500000

258;Blanco Domingo;13;Med;3500000

259;Domínguez Nery;13;Med;1500000

260;Erviti Walter;13;Med;1500000

261;Meza Maximiliano;13;Med;2500000

262;Rigoni Emiliano;13;Med;3500000

59;Molina Emanue;13;Med;3500000

1;Equipo01;Usuario01

2;Equipo02;Usuario02

3;Equipo03;Usuario03

4;Equipo04;Usuario04

5;Equipo05;Usuario05

6;Equipo06;Usuario06

7;Equipo07;Usuario07

8;Equipo08;Usuario08

9;Equipo09;Usuario09

10;Equipo10;Usuario1

ValorMaximoXEquipo=20000000

ARQ=1

DEF=2

MED=2

DEL=1

1;1

1;2

2;24

2;3

2;4

2;10

2;247

3;169

3;235

3;169

3;62

3;6

4;44

4;71

6;124

4;72

4;184

4;37

4;38

5;66

5;3

5;4

5;5

6;3

6;4

6;183

6;38

6;39

7;39

7;300

1;1;5

2;1;4

24;1;7

3;1;3

4;1;8

10;1;6

247;1;2

169;1;5

235;1;9

62;1;4

6;1;2

44;1;7

266;1;6

71;1;3

124;1;2

72;1;6

184;1;7

37;1;9

38;1;1

66;1;5

5;1;3

183;1;7

39;1;8

1;2;6

266;2;1

2;2;3

24;2;6

3;2;9

4;2;3

10;2;9

247;2;4

169;2;2

235;2;7

62;2;8

6;2;4

44;2;5

71;2;2

124;2;7

72;2;5

184;2;9

37;2;9

38;2;0

66;2;4

5;2;6

183;2;0

39;2;6

1;3;3

2;3;5

3;3;8

4;3;2

5;3;4

6;3;1

10;3;8

24;3;6

37;3;0

38;3;4

39;3;6

44;3;3

62;3;8

66;3;9

71;3;4

72;3;3

124;3;2

169;3;6

183;3;8

184;3;4

235;3;7

247;3;3

247;4;8

235;4;9

184;4;5

183;4;4

169;4;8

124;4;3

72;4;5

71;4;2

66;4;2

62;4;2

44;4;2

39;4;4

38;4;7

37;4;5

24;4;2

10;4;7

6;4;9

5;4;7

4;4;2

3;4;1

2;4;7

1;4;9