#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <cstring>

#include <cstdlib>

#include <stdio.h>

#include "Equipo.h"

#include "ListaEquipos.h"

#include "Club.h"

#include "Listaclubes.h"

#include "Jugador.h"

#include "ListaJugador.h"

#include "Puntajes.h"

#include "ListaPuntajes.h"

#include "ListaPuntajesTotal.h"

#include "Error.h"

#include "ListaErrores.h"

#include "ListaPuntajesEquipoPorFecha.h"

using namespace std;

//LEER

void leer\_archivoConfiguracion();

void leer\_archivoPuntajes(ListaPuntajes &puntajes);

void leer\_archivoEquipos(ListaEquipo &equipos);

void leer\_archivoJugador(ListaJugador &jugadores);

void leer\_archivoJugadoresPorEquipo(ListaEquipo &equipos,ListaJugador &jugadores);

void leer\_archivoClubes(ListaClubes &clubes);

//VALIDACIONES

void validarEquipos(ListaEquipo &listaequipo);

void validarJugadores(ListaJugador &listajugador,ListaClubes &listaclubes);

//FUNCIONES VARIAS

void asignarClubes(ListaClubes &clubes, ListaJugador &jugadores);

void ordenarListaPuntajesTotal(ListaPuntajesTotal&lista);

ResultadoComparacionPuntajesTotal compararDatoPuntajesTotal(PuntajesTotal puntaje1, PuntajesTotal puntaje2);/\*lo declaro para usarlo en la comparacion\*/

bool buscarJugadorEnEquipo(Equipo &equipo, int idjugador);

Jugador traerJugador(ListaJugador &jugadores,int idjugador);

void asignarJugadorAEquipo(ListaEquipo &equipos, Jugador jugador, int idequipo);

void imprimir\_listaEquipos(ListaEquipo &equipos);

void crear\_archivo\_puntajesPorJugador(ListaPuntajesTotal&lista);

Equipo traerEquipo(ListaEquipo &equipos,int idequipo);

void puntajesEquiposPorFecha(ListaEquipo &equipos,ListaPuntajes &puntajes);

void asignarPuntosAEquipos(ListaPuntajesEquipoPorFecha &resultado, ListaEquipo &equipos, int idjugador, float puntos);

void asignarPuntosEnEquipoPorFecha(ListaPuntajesEquipoPorFecha &listapuntajes, Equipo equipo, float puntos);

void crear\_archivo\_puntajesEquiposPorFecha(ListaPuntajesEquipoPorFecha &resultado, int fecha);

void ordenarListaPuntajesEquipoPorFecha(ListaPuntajesEquipoPorFecha&lista);

void crear\_archivo\_puntajesEquiposTotal(ListaPuntajesEquipoPorFecha &resultado);

void puntajesEquiposTotal(ListaEquipo &equipos, ListaPuntajes &puntajes);

void menu\_carga(ListaClubes &clubes, ListaJugador &jugadores, ListaEquipo &equipo);

void menu\_procesamiento(ListaJugador &jugadores, ListaEquipo &equipos, ListaPuntajes &puntajes, ListaPuntajesTotal &listatotal);

void puntosPorJugadorTotal(ListaPuntajes&lista1,ListaJugador&lista2, ListaPuntajesTotal&lista3);

//CONFIGURACION

int valorMaximoXEquipo;

int valorARQ;

int valorDEF;

int valorMED;

int valorDEL;

int main()

{

//CARGA ARCHIVOS

//ARCHIVO CONFIGURACION

leer\_archivoConfiguracion();

//LISTAS MAESTRAS

ListaJugador jugadores;

crearListaJugador(jugadores);

ListaClubes clubes;

crearListaClubes(clubes);

ListaEquipo equipos;

crearListaEquipo(equipos);

ListaPuntajes puntajes;

crearListaPuntajes(puntajes);

ListaPuntajesTotal listatotal;

crearListaPuntajesTotal(listatotal);

//LECTURAS-CARGAS

leer\_archivoPuntajes(puntajes);

//CARACTERES SPANISH

setlocale(LC\_ALL,"Spanish");

//MENU PRINCIPAL

int opcion=0;

while(opcion!=8){

system("cls");

cout << "Gran DT" << endl << "-----------------------" << endl << "1- Menú carga de datos" << endl << "2- Menú procesamiento de fechas" << endl << "8- Salir" << endl << endl << "Ingrese una opción: ";

cin >> opcion;

switch(opcion){

case 1: menu\_carga(clubes,jugadores,equipos); break;

case 2: menu\_procesamiento(jugadores,equipos,puntajes,listatotal); break;

default: break;

}

}

//FIN

return 0;

}

void asignarClubes(ListaClubes &clubes, ListaJugador &jugadores){

//TDA

Clubes club;

crear(club);

Jugador jugador;

constructor(jugador);

ListaJugador jugadores\_club;

crearListaJugador(jugadores\_club);

//cursor clubes

PtrNodoListaClubes cursorclub;

cursorclub=primero(clubes);

//cursor jugadores

PtrNodoListaJugador cursorjugador;

cursorjugador=primeroListaJugador(jugadores);

while(cursorclub!=finClubes()){

obtenerDato(clubes,club,cursorclub);

jugadores\_club=getLista(club);

//RECORRER LISTA DE JUGADORES

while(cursorjugador!=finListaJugador()){

obtenerDatoListaJugador(jugadores,jugador,cursorjugador);

//SI ES EL MISMO ID QUE EL CLUB ASIGNA AL JUGADOR A LA LISTA DEL CLUB

if(getClub(jugador)==getId(club)){

adicionarFinalListaJugador(jugadores\_club,jugador); //CAMBIAR POR GET

}

cursorjugador=siguienteListaJugador(jugadores,cursorjugador);

}

setListaJugadores(club,jugadores\_club);

colocarDato(clubes,club,cursorclub);

cursorjugador=primeroListaJugador(jugadores);

cursorclub=siguiente(clubes,cursorclub);

}

}

void leer\_archivoConfiguracion(){

ifstream archivo;

std::string str,variable,valor\_string,var;

int valor;

archivo.open("configuracion.conf",ios::in);

if(archivo.fail()){

cout << "No se pudo abrir el archivo" << endl;

}

else {

while(getline(archivo,variable,'=')&&getline(archivo,valor\_string)){

valor= atoi(valor\_string.c\_str());

if(variable=="ValorMaximoXEquipo"){

valorMaximoXEquipo=valor;

}

else if(variable=="ARQ"){

valorARQ=valor;

}

else if(variable=="DEF"){

valorDEF=valor;

}

else if(variable=="MED"){

valorMED=valor;

}

else if(variable=="DEL"){

valorDEL=valor;

}

}

}

archivo.close();

}

void leer\_archivoPuntajes(ListaPuntajes &puntajes){

Puntajes puntaje;

crear(puntaje);

string idjugador,fecha,puntos;

ifstream archivo;

archivo.open("PuntosPorJugador.db",ios::in);

if(archivo.fail()){

cout << "ERROR!: No se pudo abrir el archivo" << endl; //AGREGAR A ERRORES?

}

else {

while(getline(archivo,idjugador,';')&&getline(archivo,fecha,';')&&getline(archivo,puntos)){

setId(puntaje,atoi(idjugador.c\_str()));

setFecha(puntaje,atoi(fecha.c\_str()));

setPuntos(puntaje,atof(puntos.c\_str()));

adicionarFinal(puntajes,puntaje);

}

}

archivo.close();

destruir(puntaje);

}

void leer\_archivoEquipos(ListaEquipo &equipos){

Equipo equipo;

crear(equipo);

ListaJugador jugadores;

crearListaJugador(jugadores);

string id,nomequipo,nomusuario;

ifstream archivo;

archivo.open("Equipos.db",ios::in);

if(archivo.fail()){

cout << "ERROR!: No se pudo abrir el archivo" << endl; //AGREGAR A ERRORES?

}

else {

while(getline(archivo,id,';')&&getline(archivo,nomequipo,';')&&getline(archivo,nomusuario)){

setID(equipo,atoi(id.c\_str()));

setNombreEquipo(equipo,nomequipo);

setNombreUsuario(equipo,nomusuario);

setListaArqueros(equipo,jugadores);

setListaDefensores(equipo,jugadores);

setListaMediocampistas(equipo,jugadores);

setListaDelanteros(equipo,jugadores);

adicionarFinal(equipos,equipo);

}

}

archivo.close();

destruir(equipo);

eliminarListaJugador(jugadores);

}

void leer\_archivoJugador(ListaJugador &jugadores){

Jugador jugador;

constructor(jugador);

string id,nombre,club,posicion,valor;

ifstream archivo;

archivo.open("Jugadores.db",ios::in);

if(archivo.fail()){

cout << "ERROR!: No se pudo abrir el archivo" << endl; // AGREGAR AL ARCHIVO ERRORES?

}

else {

while(getline(archivo,id,';')&&getline(archivo,nombre,';')&&getline(archivo,club,';')&&getline(archivo,posicion,';')&&getline(archivo,valor)){

setId(jugador,atoi(id.c\_str()));

setClub(jugador,atoi(club.c\_str()));

setNombre(jugador,nombre);

if(posicion=="ARQ"){

setPosicion(jugador,ARQ);

}

if(posicion=="DEF"){

setPosicion(jugador,DEF);

}

if(posicion=="MED"){

setPosicion(jugador,MED);

}

if(posicion=="DEL"){

setPosicion(jugador,DEL);

}

setValor(jugador,atof(valor.c\_str()));

adicionarFinalListaJugador(jugadores, jugador);

}

}

archivo.close();

destructor(jugador);

}

void leer\_archivoJugadoresPorEquipo(ListaEquipo &equipos,ListaJugador &jugadores){

Equipo equipo;

Jugador jugador;

string idequipo,idjugador;

int idequipo\_int,idjugador\_int;

ifstream archivo;

archivo.open("JugadoresPorEquipo.db",ios::in);

if(archivo.fail()){

cout << "ERROR!: No se pudo abrir el archivo" << endl; //AGREGAR A ERRORES?

}

else {

// LEER REGISTRO DEL ARCHIVO

while(getline(archivo,idequipo,';')&&getline(archivo,idjugador)){

idequipo\_int=atoi(idequipo.c\_str());

idjugador\_int=atoi(idjugador.c\_str());

// TRAER JUGADOR POR ID

jugador=traerJugador(jugadores,idjugador\_int);

// ASIGNAR JUGADOR A EQUIPO

asignarJugadorAEquipo(equipos,jugador,idequipo\_int);

}

}

archivo.close();

destruir(equipo);

destructor(jugador);

}

Jugador traerJugador(ListaJugador &jugadores, int idjugador){

PtrNodoListaJugador cursor;

cursor=primeroListaJugador(jugadores);

Jugador jugador;

constructor(jugador);

Jugador resultado;

constructor(resultado);

bool encontrado=false;

while(cursor!=finListaJugador()&&!encontrado){

obtenerDatoListaJugador(jugadores,jugador,cursor);

if(idjugador==getId(jugador)){

encontrado=true;

resultado=jugador;

}

cursor=siguienteListaJugador(jugadores,cursor);

}

destructor(jugador);

return resultado;

}

Equipo traerEquipo(ListaEquipo &equipos,int idequipo){

PtrNodoListaEquipo cursor;

cursor=primeroEquipo(equipos);

Equipo equipo;

crear(equipo);

Equipo resultado;

crear(equipo);

bool encontrado=false;

while(cursor!=finEquipo()&&!encontrado){

obtenerDato(equipos,equipo,cursor);

if(idequipo==getID(equipo)){

encontrado=true;

resultado=equipo;

}

cursor=siguienteEquipo(equipos,cursor);

}

destruir(equipo);

return resultado;

}

void asignarJugadorAEquipo(ListaEquipo &equipos, Jugador jugador, int idequipo){

PtrNodoListaEquipo cursor;

cursor=primeroEquipo(equipos);

Equipo equipo,

crear(equipo);

ListaJugador jugadores1;

crearListaJugador(jugadores1);

bool encontrado=false;

Posicion posicion;

posicion=getPosicion(jugador);

while(cursor!=finEquipo()&&!encontrado){

obtenerDato(equipos,equipo,cursor);

if(getID(equipo)==idequipo){

encontrado=true;

if(posicion==ARQ){

eliminarListaJugador(jugadores1);

crearListaJugador(jugadores1);

jugadores1=getListaArqueros(equipo);

adicionarFinalListaJugador(jugadores1,jugador);

setListaArqueros(equipo,jugadores1);

}

if(posicion==DEF){

eliminarListaJugador(jugadores1);

crearListaJugador(jugadores1);

jugadores1=getListaDefensores(equipo);

adicionarFinalListaJugador(jugadores1,jugador);

setListaDefensores(equipo,jugadores1);

}

if(posicion==MED){

eliminarListaJugador(jugadores1);

crearListaJugador(jugadores1);

jugadores1=getListaMediocampistas(equipo);

adicionarFinalListaJugador(jugadores1,jugador);

setListaMediocampistas(equipo,jugadores1);

}

if(posicion==DEL){

eliminarListaJugador(jugadores1);

crearListaJugador(jugadores1);

jugadores1=getListaDelanteros(equipo);

adicionarFinalListaJugador(jugadores1,jugador);

setListaDelanteros(equipo,jugadores1);

}

colocarDato(equipos,equipo,cursor);

}

cursor=siguienteEquipo(equipos,cursor);

}

destruir(equipo);

destructor(jugador);

// eliminarListaJugador(jugadores1);

}

void leer\_archivoClubes(ListaClubes &clubes){

Clubes club;

ListaJugador jugadores;

string idclub,nombre;

ifstream archivo;

archivo.open("Clubes.db",ios::in);

if(archivo.fail()){

cout << "ERROR!: No se pudo abrir el archivo" << endl; //AGREGAR A ERRORES?

}

else {

// LEER REGISTRO DEL ARCHIVO

while(getline(archivo,idclub,';')&&getline(archivo,nombre)){

setId(club,atoi(idclub.c\_str()));

setNombre(club,nombre);

crearListaJugador(jugadores);

setListaJugadores(club,jugadores);

adicionarFinal(clubes,club);

}

}

archivo.close();

destruir(club);

eliminarListaJugador(jugadores);

}

void imprimir\_listaEquipos(ListaEquipo &equipos){

PtrNodoListaEquipo cursor;

cursor=primeroEquipo(equipos);

PtrNodoListaJugador cursorjugador;

Equipo equipo;

crear(equipo);

Jugador jugador;

constructor(jugador);

ListaJugador jugadores;

crearListaJugador(jugadores);

while(cursor!=finEquipo()){

obtenerDato(equipos,equipo,cursor);

//IMPRIMIR DATOS EQUIPO

cout << "ID: " << getID(equipo) << " Nombre " << getNombreEquipo(equipo) << " Usuario: " << getNombreUsuario(equipo) << endl;

//IMPRIMIR LISTA ARQUEROS

jugadores=getListaArqueros(equipo);

cursorjugador=primeroListaJugador(jugadores);

while(cursorjugador!=finListaJugador()){

obtenerDatoListaJugador(jugadores,jugador,cursorjugador);

cout << "Jugador- ID: " << getId(jugador) << " Nombre: " << getNombre(jugador) << endl;

cursorjugador=siguienteListaJugador(jugadores,cursorjugador);

}

//IMPRIMIR LISTA DEFENSORES

jugadores=getListaDefensores(equipo);

cursorjugador=primeroListaJugador(jugadores);

while(cursorjugador!=finListaJugador()){

obtenerDatoListaJugador(jugadores,jugador,cursorjugador);

cout << "Jugador- ID: " << getId(jugador) << " Nombre: " << getNombre(jugador) << endl;

cursorjugador=siguienteListaJugador(jugadores,cursorjugador);

}

//IMPRIMIR LISTA MEDIOS

jugadores=getListaMediocampistas(equipo);

cursorjugador=primeroListaJugador(jugadores);

while(cursorjugador!=finListaJugador()){

obtenerDatoListaJugador(jugadores,jugador,cursorjugador);

cout << "Jugador- ID: " << getId(jugador) << " Nombre: " << getNombre(jugador) << endl;

cursorjugador=siguienteListaJugador(jugadores,cursorjugador);

}

//IMPRIMIR LISTA DELANTEROS

jugadores=getListaDelanteros(equipo);

cursorjugador=primeroListaJugador(jugadores);

while(cursorjugador!=finListaJugador()){

obtenerDatoListaJugador(jugadores,jugador,cursorjugador);

cout << "Jugador- ID: " << getId(jugador) << " Nombre: " << getNombre(jugador) << endl;

cursorjugador=siguienteListaJugador(jugadores,cursorjugador);

}

cursor=siguienteEquipo(equipos,cursor);

}

eliminarListaJugador(jugadores);

destructor(jugador);

destruir(equipo);

}

void crear\_archivo\_puntajesPorJugador(ListaPuntajesTotal&lista){

PtrNodoListaPuntajesTotal cursor;

cursor=primero(lista);

int posicion=0;

PuntajesTotal puntajes; /\*tipo de dato definido dentro del tda listaPuntajesTotal\*/

PuntajesTotal puntajes1;

bool primero=true;

/\*creo el archivo que pide\*/

FILE\*archi;

archi=fopen("puntajesPorJugador.txt","w");

while(cursor!=finPuntajesTotal()){

/\*le asigno la posicion\*/

obtenerDatoPuntajesTotal(lista,puntajes,cursor);

obtenerDatoPuntajesTotal(lista,puntajes1,anterior(lista,cursor));

if(primero==true)

posicion=1;

if((puntajes.puntaje!=puntajes1.puntaje)&&(primero==false)){

posicion++;

puntajes.posicion=posicion;

colocarDatoPuntajesTotal(lista,puntajes,cursor);

}

else{

puntajes.posicion=posicion;

colocarDatoPuntajesTotal(lista,puntajes,cursor);

}

/\*vuelco al archivo\*/

fprintf(archi,"%d;%d;%d\n",puntajes.posicion,puntajes.idJugador,(int)puntajes.puntaje);

cursor=siguiente(lista,cursor);

primero=false;

}

fclose(archi);

}

void ordenarListaPuntajesTotal(ListaPuntajesTotal&lista)/\*de mayor a menor, de acuerdo al criterio de resultadocomparacion\*/{

PtrNodoListaPuntajesTotal cursor;

PtrNodoListaPuntajesTotal cursorsgte;

PuntajesTotal puntaje1;

PuntajesTotal puntaje2;

bool encontrado=false;

while(encontrado!=true){

encontrado=true;

cursor=primero(lista);

cursorsgte=siguiente(lista,cursor);

while(cursor!=finPuntajesTotal())

{

obtenerDatoPuntajesTotal(lista,puntaje1,cursor);

obtenerDatoPuntajesTotal(lista,puntaje2,cursorsgte);

if(compararDatoPuntajesTotal(puntaje2,puntaje1)==MAYORPUNTAJETOTAL)

{

colocarDatoPuntajesTotal(lista,puntaje1,cursorsgte);

colocarDatoPuntajesTotal(lista,puntaje2,cursor);

encontrado=false;

cursor=primero(lista);

cursorsgte=siguiente(lista,cursor);

}

else

{

cursor=siguiente(lista,cursor);

cursorsgte=siguiente(lista,cursor);

}

}

}

}

void validarEquipos(ListaEquipo &listaequipo,ListaClubes &listaclubes){

float valorTotal=0;

int arqueros=0;

int defensores=0;

int mediocampistas=0;

int delanteros=0;

bool errorEncontrado=false;

ListaError listaerror;

Error error;

Equipo equipo;

Jugador jugador;

crearListaError(listaerror);

constructor(jugador);

crear(error);

crear(equipo);

PtrNodoListaEquipo cursor;

cursor=primeroEquipo(listaequipo);

while(cursor!=finEquipo())

{

obtenerDato(listaequipo,equipo,cursor);

///OBTENGO LA CANTIDAD DE ARQUEROS POR EQUIPO Y LA SUMA DE SUS VALORES///

validarJugadores(equipo.listaarqueros,listaclubes);

PtrNodoListaJugador cursorARQ;

cursorARQ=primeroListaJugador(equipo.listaarqueros);

while(cursorARQ!=finListaJugador())

{

obtenerDatoListaJugador(equipo.listaarqueros,jugador,cursorARQ);

valorTotal=valorTotal+getValor(jugador);

cursorARQ=siguienteListaJugador(equipo.listaarqueros,cursorARQ);

}

arqueros=longitudListaJugador(equipo.listaarqueros);

///OBTENER LA CANTIDAD DE DEFENSORES POR EQUIPO Y LA SUMA DE SUS VALORES//

validarJugadores(equipo.listadefensores,listaclubes);

PtrNodoListaJugador cursorDEF;

cursorDEF=primeroListaJugador(equipo.listadefensores);

while(cursorDEF!=finListaJugador())

{

obtenerDatoListaJugador(equipo.listadefensores,jugador,cursorDEF);

valorTotal=valorTotal+getValor(jugador);

cursorDEF=siguienteListaJugador(equipo.listadefensores,cursorDEF);

}

defensores=longitudListaJugador(equipo.listadefensores);

///OBTENER LA CANTIDAD DE MEDIOCAMPISTAS Y LA SUMA DE SUS VALORES//

validarJugadores(equipo.listamedios,listaclubes);

PtrNodoListaJugador cursorMED;

cursorMED=primeroListaJugador(equipo.listamedios);

while(cursorMED!=finListaJugador())

{

obtenerDatoListaJugador(equipo.listamedios,jugador,cursorMED);

valorTotal=valorTotal+getValor(jugador);

cursorMED=siguienteListaJugador(equipo.listamedios,cursorMED);

}

mediocampistas=longitudListaJugador(equipo.listamedios);

///OBTENER LA CANTIDAD DE DELANTEROS Y LA SUMA DE SUS VALORES//

validarJugadores(equipo.listadelanteros,listaclubes);

PtrNodoListaJugador cursorDEL;

cursorDEL=primeroListaJugador(equipo.listadelanteros);

while(cursorDEL!=finListaJugador())

{

obtenerDatoListaJugador(equipo.listadelanteros,jugador,cursorDEL);

valorTotal=valorTotal+getValor(jugador);

cursorDEL=siguienteListaJugador(equipo.listadelanteros,cursorDEL);

}

delanteros=longitudListaJugador(equipo.listadelanteros);

///VALIDAR EL EQUIPO//

if(valorTotal>valorMaximoXEquipo)

{

setCodigoError(error,2);

adicionarFinal(listaerror,error);

errorEncontrado=true;

}

if(arqueros>valorARQ)

{

setCodigoError(error,3);

adicionarFinal(listaerror,error);

errorEncontrado=true;

}

if(defensores>valorDEF)

{

setCodigoError(error,4);

adicionarFinal(listaerror,error);

errorEncontrado=true;

}

if(mediocampistas>valorMED)

{

setCodigoError(error,5);

adicionarFinal(listaerror,error);

errorEncontrado=true;

}

if(delanteros>valorDEL)

{

setCodigoError(error,6);

adicionarFinal(listaerror,error);

errorEncontrado=true;

}

if(errorEncontrado==true)///SI TIENE UNO O MAS ERRORES LO ELIMINO DE LA LISTA///

{

eliminarNodoEquipo(listaequipo,cursor);

}

arqueros=0;

delanteros=0;

mediocampistas=0;

defensores=0;

valorTotal=0;

errorEncontrado=false;

cursor=siguienteEquipo(listaequipo,cursor);

}

///GUARDAR EN UN CPP TODOS LOS ERRORES ENCONTRADOS/////

GuardarDatos(error,listaerror);

eliminarListaError(listaerror);

destructor(jugador);

destruir(error);

destruir(equipo);

}

void validarJugadores(ListaJugador &listajugador,ListaClubes &listaclubes){

Jugador jugador;

Clubes club;

ListaError listaerror;

Error error;

crear(club);

crear(error);

constructor(jugador);

crearListaError(listaerror);

bool encontrado=false;

PtrNodoListaJugador cursorJugador;

PtrNodoListaClubes cursorClub;

cursorJugador=primeroListaJugador(listajugador);

while(cursorJugador!=finListaJugador())

{

obtenerDatoListaJugador(listajugador,jugador,cursorJugador);

cursorClub=primero(listaclubes);

while(cursorClub!=finClubes() && encontrado==false)

{

obtenerDato(listaclubes,club,cursorClub);

if(getId(club)==getClub(jugador))

{

encontrado=true;

}

cursorClub=siguiente(listaclubes,cursorClub);

}

if(!encontrado)

{

eliminarNodoListaJugador(listajugador,cursorJugador);

setCodigoError(error,1);

adicionarFinal(listaerror,error);

}

else encontrado=false;

cursorJugador=siguienteListaJugador(listajugador,cursorJugador);

}

GuardarDatos(error,listaerror);

eliminarListaError(listaerror);

destruir(club);

destruir(error);

destructor(jugador);

}

void ordenarListaPuntajesEquipoPorFecha(ListaPuntajesEquipoPorFecha &lista)/\*de mayor a menor, de acuerdo al criterio de resultadocomparacion\*/{

PtrNodoListaPuntajesEquipoPorFecha cursor;

PtrNodoListaPuntajesEquipoPorFecha cursorsgte;

PuntajesEquipoPorFecha puntaje1;

PuntajesEquipoPorFecha puntaje2;

bool encontrado=false;

while(encontrado!=true){

encontrado=true;

cursor=primero(lista);

cursorsgte=siguiente(lista,cursor);

while(cursor!=finPuntajesEquipoPorFecha())

{

obtenerDatoPuntajesTotal(lista,puntaje1,cursor);

obtenerDatoPuntajesTotal(lista,puntaje2,cursorsgte);

if(puntaje2.puntos>puntaje1.puntos)

{

colocarDatoPuntajesTotal(lista,puntaje1,cursorsgte);

colocarDatoPuntajesTotal(lista,puntaje2,cursor);

encontrado=false;

cursor=primero(lista);

cursorsgte=siguiente(lista,cursor);

}

else

{

cursor=siguiente(lista,cursor);

cursorsgte=siguiente(lista,cursor);

}

}

}

}

void puntajesEquiposPorFecha(ListaEquipo &equipos,ListaPuntajes &puntajes){

PtrNodoListaPuntajes cursorpuntajes;

cursorpuntajes=primero(puntajes);

Puntajes puntaje;

crear(puntaje);

ListaPuntajesEquipoPorFecha resultado;

crearListaPuntajesTotal(resultado);

int fecha=1;

while(cursorpuntajes!=finPuntajes()){

obtenerDatoPuntajes(puntajes,puntaje,cursorpuntajes);

if(getFecha(puntaje)==fecha){

//SUMARLE LOS PUNTOS A CADA EQUIPO QUE TENGA ESTE JUGADOR

asignarPuntosAEquipos(resultado,equipos,getId(puntaje),getPuntos(puntaje));

} else {

// ORDENAR LISTA DE EQUIPOS POR FECHA

ordenarListaPuntajesEquipoPorFecha(resultado);

// IMPRIMIR EL ARCHIVO DE SALIDA DE LA FECHA

crear\_archivo\_puntajesEquiposPorFecha(resultado,fecha);

crearListaPuntajesTotal(resultado);

asignarPuntosAEquipos(resultado,equipos,getId(puntaje),getPuntos(puntaje));

fecha=getFecha(puntaje);

}

cursorpuntajes=siguiente(puntajes,cursorpuntajes);

}

crear\_archivo\_puntajesEquiposPorFecha(resultado,fecha);

destruir(puntaje);

}

void crear\_archivo\_puntajesEquiposPorFecha(ListaPuntajesEquipoPorFecha &resultado,int fecha){

ofstream archivo;

string nombre\_archivo,prefijo,fecha\_string,tipo;

char intStr[2];

itoa(fecha,intStr,10);

fecha\_string = string(intStr);

tipo=".txt";

prefijo="puntajesFecha";

nombre\_archivo=prefijo+fecha\_string+tipo;

bool pri=true;

int posicion=0;

PtrNodoListaPuntajesEquipoPorFecha cursor;

cursor=primero(resultado);

PuntajesEquipoPorFecha puntaje;

PuntajesEquipoPorFecha puntaje1;

archivo.open(nombre\_archivo.c\_str(),ios::out);

if(archivo.fail()){

cout << "No se pudo crear el archivo" << endl;

}

else {

while(cursor!=finPuntajesEquipoPorFecha()){

obtenerDatoPuntajesTotal(resultado,puntaje,cursor);

obtenerDatoPuntajesTotal(resultado,puntaje1,anterior(resultado,cursor));

if(pri==true)

posicion=1;

if((puntaje.puntos!=puntaje1.puntos)&&(pri==false))

posicion++;

archivo << posicion << ";" << puntaje.idEquipo << ";" << puntaje.nombreequipo << ";" << puntaje.nombreusuario << ";" << puntaje.puntos << endl;

cursor=siguiente(resultado,cursor);

pri=false;

}

archivo.close();

}

}

bool buscarJugadorEnEquipo(Equipo &equipo, int idjugador){

bool resultado=false;

ListaJugador jugadores;

crearListaJugador(jugadores);

Jugador jugador;

constructor(jugador);

PtrNodoListaJugador cursor;

jugadores=getListaArqueros(equipo);

cursor=primeroListaJugador(jugadores);

while(cursor!=finListaJugador()&&!resultado){

obtenerDatoListaJugador(jugadores,jugador,cursor);

if(getId(jugador)==idjugador){

resultado=true;

}

cursor=siguienteListaJugador(jugadores,cursor);

}

jugadores=getListaDefensores(equipo);

cursor=primeroListaJugador(jugadores);

while(cursor!=finListaJugador()&&!resultado){

obtenerDatoListaJugador(jugadores,jugador,cursor);

if(getId(jugador)==idjugador){

resultado=true;

}

cursor=siguienteListaJugador(jugadores,cursor);

}

jugadores=getListaMediocampistas(equipo);

cursor=primeroListaJugador(jugadores);

while(cursor!=finListaJugador()&&!resultado){

obtenerDatoListaJugador(jugadores,jugador,cursor);

if(getId(jugador)==idjugador){

resultado=true;

}

cursor=siguienteListaJugador(jugadores,cursor);

}

jugadores=getListaDelanteros(equipo);

cursor=primeroListaJugador(jugadores);

while(cursor!=finListaJugador()&&!resultado){

obtenerDatoListaJugador(jugadores,jugador,cursor);

if(getId(jugador)==idjugador){

resultado=true;

}

cursor=siguienteListaJugador(jugadores,cursor);

}

return resultado;

}

void asignarPuntosAEquipos(ListaPuntajesEquipoPorFecha &resultado, ListaEquipo &equipos, int idjugador, float puntos){

PtrNodoListaEquipo cursor;

cursor=primeroEquipo(equipos);

Equipo equipo;

crear(equipo);

while(cursor!=finEquipo()){

obtenerDato(equipos,equipo,cursor);

if(buscarJugadorEnEquipo(equipo,idjugador)==1){

// SI EL JUGADOR ESTABA EN ESE EQUIPO ASIGNARLE LOS PUNTOS Y SUBIRLO A LA LISTA DE ESA FECHA

asignarPuntosEnEquipoPorFecha(resultado,equipo,puntos);

}

cursor=siguienteEquipo(equipos,cursor);

}

}

void asignarPuntosEnEquipoPorFecha(ListaPuntajesEquipoPorFecha &listapuntajes, Equipo equipo, float puntos){

PtrNodoListaPuntajesEquipoPorFecha cursor;

cursor=primero(listapuntajes);

bool encontrado=false;

PuntajesEquipoPorFecha puntajeequipoporfecha;

while(cursor!=finPuntajesEquipoPorFecha()&&!encontrado){

obtenerDatoPuntajesTotal(listapuntajes,puntajeequipoporfecha,cursor);

if(puntajeequipoporfecha.idEquipo==getID(equipo)){

//SI EL EQUIPO YA ESTA EN LA LISTA SUMARLE LOS PUNTOS A ESE NODO

puntajeequipoporfecha.puntos=puntajeequipoporfecha.puntos+puntos;

colocarDatoPuntajesTotal(listapuntajes,puntajeequipoporfecha,cursor);

encontrado=true;

}

cursor=siguiente(listapuntajes,cursor);

}

//SI EL EQUIPO NO ESTA, ADICIONARLO CON SUS PUNTOS

if(!encontrado){

puntajeequipoporfecha.idEquipo=getID(equipo);

puntajeequipoporfecha.nombreequipo=getNombreEquipo(equipo);

puntajeequipoporfecha.nombreusuario=getNombreUsuario(equipo);

puntajeequipoporfecha.puntos=puntos;

adicionarFinal(listapuntajes,puntajeequipoporfecha);

}

}

void puntajesEquiposTotal(ListaEquipo &equipos, ListaPuntajes &puntajes){

PtrNodoListaPuntajes cursorpuntajes;

cursorpuntajes=primero(puntajes);

Puntajes puntaje;

crear(puntaje);

ListaPuntajesEquipoPorFecha resultado;

crearListaPuntajesTotal(resultado);

while(cursorpuntajes!=finPuntajes()){

obtenerDatoPuntajes(puntajes,puntaje,cursorpuntajes);

//SUMARLE LOS PUNTOS A CADA EQUIPO QUE TENGA ESTE JUGADOR

asignarPuntosAEquipos(resultado,equipos,getId(puntaje),getPuntos(puntaje));

cursorpuntajes=siguiente(puntajes,cursorpuntajes);

}

// ORDENAR LISTA DE EQUIPOS

ordenarListaPuntajesEquipoPorFecha(resultado);

// IMPRIMIR EL ARCHIVO DE SALIDA DE LA FECHA

crear\_archivo\_puntajesEquiposTotal(resultado);

destruir(puntaje);

}

void crear\_archivo\_puntajesEquiposTotal(ListaPuntajesEquipoPorFecha &resultado){

ofstream archivo;

int posicion=0;

bool prim=true;

PtrNodoListaPuntajesEquipoPorFecha cursor;

cursor=primero(resultado);

PuntajesEquipoPorFecha puntaje1;

PuntajesEquipoPorFecha puntaje;

archivo.open("puntajesTotalesPorEquipo.txt",ios::out);

while(cursor!=finPuntajesEquipoPorFecha()){

obtenerDatoPuntajesTotal(resultado,puntaje,cursor);

obtenerDatoPuntajesTotal(resultado,puntaje1,anterior(resultado,cursor));

if(prim==true)

posicion=1;

if((puntaje.puntos!=puntaje1.puntos)&&(prim==false))

posicion++;

archivo << posicion << ";" << puntaje.idEquipo << ";" << puntaje.nombreequipo << ";" << puntaje.nombreusuario << ";" << puntaje.puntos << endl;

cursor=siguiente(resultado,cursor);

prim=false;

}

archivo.close();

}

void menu\_carga(ListaClubes &clubes, ListaJugador &jugadores, ListaEquipo &equipos){

system("cls");

int opcion=0;

while(opcion!=8){

system("cls");

cout << "Menú de carga de datos" << endl << "-----------------------" << endl << "1- Cargar clubes.db" << endl << "2- Cargar jugadores.db" << endl << "3- Cargar equipos.db" << endl << "4- Cargar jugadoresPorEquipo.db" << endl << "8- Volver" << endl << endl << "Ingrese una opción: ";

cin >> opcion;

switch(opcion){

case 1:

leer\_archivoClubes(clubes);

cout << endl << "-----------------------" << endl << "Archivo clubes.db cargado con éxito" << endl << endl;

system("pause");

break;

case 2:

leer\_archivoJugador(jugadores);

asignarClubes(clubes,jugadores);

cout << endl << "-----------------------" << endl << "Archivo jugadores.db cargado con éxito" << endl << endl;

system("pause");

break;

case 3:

leer\_archivoEquipos(equipos);

cout << endl << "-----------------------" << endl << "Archivo equipos.db cargado con éxito" << endl << endl;

system("pause");

break;

case 4:

leer\_archivoJugadoresPorEquipo(equipos,jugadores);

validarEquipos(equipos,clubes);

cout << endl << "-----------------------" << endl << "Archivo jugadoresPorEquipo.db cargado con éxito" << endl << endl;

system("pause");

break;

default:

break;

}

}

}

void menu\_procesamiento(ListaJugador &jugadores, ListaEquipo &equipos, ListaPuntajes &puntajes, ListaPuntajesTotal &listatotal){

system("cls");

int opcion=0;

while(opcion!=8){

system("cls");

cout << "Menú procesamiento de fechas" << endl << "-----------------------" << endl << "1- Puntajes por fecha" << endl << "2- Puntajes totales" << endl << "3- Puntajes totales por jugador" << endl << "8- Volver" << endl << endl << "Ingrese una opción: ";

cin >> opcion;

switch(opcion){

case 1:

//PUNTOS EQUIPOS POR FECHA

puntajesEquiposPorFecha(equipos,puntajes);

cout << endl << "-----------------------" << endl << "Archivo/s puntajesFechaXX.txt generado/s con éxito" << endl << endl;

system("pause");

break;

case 2:

//PUNTOS TOTALES POR EQUIPO

puntajesEquiposTotal(equipos,puntajes);

cout << endl << "-----------------------" << endl << "Archivo puntajesTotalesPorEquipo.txt generado con éxito" << endl << endl;

system("pause");

break;

case 3:

//PUNTOSTOTAL X JUGADOR

puntosPorJugadorTotal(puntajes,jugadores,listatotal);

ordenarListaPuntajesTotal(listatotal);

crear\_archivo\_puntajesPorJugador(listatotal);

cout << endl << "-----------------------" << endl << "Archivo puntajesPorJugador.txt generado con éxito" << endl << endl;

system("pause");

break;

default:

break;

}

}

}

void puntosPorJugadorTotal(ListaPuntajes&lista,ListaJugador&lista1,ListaPuntajesTotal&lista2){

PtrNodoListaJugador cursor1;

PtrNodoListaPuntajes cursor2;

float total=0;

bool estaEnequipo=false;

Jugador jugador;

Puntajes puntaje;

PuntajesTotal puntajetotal;

cursor1=primeroListaJugador(lista1);

/\*me para en el primer jugador de lista jugadores e itero la de puntajes. cuando aparece el jugador, sumo los puntos\*/

while(cursor1!=finListaJugador()){

obtenerDatoListaJugador(lista1,jugador,cursor1);

cursor2=primero(lista);

while(cursor2!=finPuntajes()){

obtenerDatoPuntajes(lista,puntaje,cursor2);

if(getId(jugador)==getId(puntaje)){

total=getPuntos(puntaje)+total;

cursor2=siguiente(lista,cursor2);

estaEnequipo=true;/\*lo uso para ver que el jugador este en la lista de puntajes\*/

}

else

cursor2=siguiente(lista,cursor2);

}

if(estaEnequipo==true){/\*si entro al menos una vez al anterior, es porque aparece y entonces sumo\*/

puntajetotal.idJugador=getId(jugador);

puntajetotal.puntaje=total;

adicionarPrincipio(lista2,puntajetotal);

total=0;

cursor1=siguienteListaJugador(lista1,cursor1);

estaEnequipo=false;

}

else

cursor1=siguienteListaJugador(lista1,cursor1);

}

}