/\* TDA Lista

\* Implementación Simplemente Enlazada

\* Archivo : ListaPuntajesPorFecha.cpp

\* Versión : 1.1

\*/

#include "ListaPuntajesPorFecha.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Definición de Tipos de Datos para manejo interno \*/

/\*--------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Funciones Adicionales \*/

/\*-----------------------\*/

/\*

pre : ninguna.

post: compara ambos dato1 y dato2, devuelve

mayor si dato1 es mayor que dato2,

igual si dato1 es igual a dato2,

menor si dato1 es menor que dato2.

puntaje1 : dato a comparar.

puntaje2 : dato a comparar.

return resultado de comparar dato1 respecto de dato2.

\*/

ResultadoComparacionPuntajesPorFecha compararDatoPuntajesPorFecha(PuntajesPorFecha dato1, PuntajesPorFecha dato2) {

if ((dato1.puntaje) >(dato2.puntaje)) {

return MAYORPUNTAJEPORFECHA;

}

else if (dato1.puntaje < dato2.puntaje) {

return MENORPUNTAJEPORFECHA;

}

else {

return IGUALPUNTAJEPORFECHA;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Implementación de Primitivas \*/

/\*------------------------------\*/

void crearListaPuntajesPorFecha(ListaPuntajesPorFecha &lista) {

lista.primero = finPuntajesPorFecha();

}

/\*----------------------------------------------------------------------------\*/

bool listaVaciaPuntajesPorFecha(ListaPuntajesPorFecha &lista) {

return (primero(lista) == finPuntajesPorFecha());

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha finPuntajesPorFecha() {

return NULL;

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha primero(ListaPuntajesPorFecha &lista) {

return lista.primero;

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha siguiente(ListaPuntajesPorFecha &lista, PtrNodoListaPuntajesPorFecha ptrNodo) {

/\* verifica si la lista está vacia o si ptrNodo es el último \*/

if ((! listaVaciaPuntajesPorFecha(lista)) && (ptrNodo != finPuntajesPorFecha()))

return ptrNodo->sgte;

else

return finPuntajesPorFecha();

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha anterior(ListaPuntajesPorFecha &lista, PtrNodoListaPuntajesPorFecha ptrNodo) {

PtrNodoListaPuntajesPorFecha ptrPrevio = finPuntajesPorFecha();

PtrNodoListaPuntajesPorFecha ptrCursor = primero(lista);

while (( ptrCursor != finPuntajesPorFecha()) && (ptrCursor != ptrNodo)) {

ptrPrevio = ptrCursor;

ptrCursor = siguiente(lista,ptrCursor);

}

return ptrPrevio;

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha ultimo(ListaPuntajesPorFecha &lista) {

/\* el último nodo de la lista es el anterior al finPuntajes() \*/

return anterior(lista,finPuntajesPorFecha());

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha crearNodoLista(PuntajesPorFecha dato) {

/\* reserva memoria para el nodo y luego completa sus datos \*/

PtrNodoListaPuntajesPorFecha ptrAux = new NodoListaPuntajesPorFecha();

ptrAux->dato = dato;

ptrAux->sgte = finPuntajesPorFecha();

return ptrAux;

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha adicionarPrincipio(ListaPuntajesPorFecha &lista, PuntajesPorFecha puntajes) {

/\* crea el nodo \*/

PtrNodoListaPuntajesPorFecha ptrNuevoNodo = crearNodoLista(puntajes);

/\* lo incorpora al principio de la lista \*/

ptrNuevoNodo->sgte = lista.primero;

lista.primero = ptrNuevoNodo;

return ptrNuevoNodo;

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha adicionarDespues(ListaPuntajesPorFecha &lista, PuntajesPorFecha puntajes, PtrNodoListaPuntajesPorFecha ptrNodo) {

PtrNodoListaPuntajesPorFecha ptrNuevoNodo = finPuntajesPorFecha();

/\* si la lista está vacia se adiciona la principio \*/

if (listaVaciaPuntajesPorFecha(lista))

ptrNuevoNodo = adicionarPrincipio(lista,puntajes);

else {

if (ptrNodo != finPuntajesPorFecha()) {

/\* crea el nodo y lo intercala en la lista \*/

ptrNuevoNodo = crearNodoLista(puntajes);

ptrNuevoNodo->sgte = ptrNodo->sgte;

ptrNodo->sgte = ptrNuevoNodo;

}

}

return ptrNuevoNodo;

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha adicionarFinal(ListaPuntajesPorFecha &lista, PuntajesPorFecha puntajes) {

/\* adiciona el dato después del último nodo de la lista \*/

return adicionarDespues(lista,puntajes,ultimo(lista));

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha adicionarAntes(ListaPuntajesPorFecha &lista, PuntajesPorFecha puntajes, PtrNodoListaPuntajesPorFecha ptrNodo) {

PtrNodoListaPuntajesPorFecha ptrNuevoNodo = finPuntajesPorFecha();

if (! listaVaciaPuntajesPorFecha(lista)) {

if (ptrNodo != primero(lista))

ptrNuevoNodo = adicionarDespues(lista,puntajes,anterior(lista,ptrNodo));

else

ptrNuevoNodo = adicionarPrincipio(lista,puntajes);

}

return ptrNuevoNodo;

}

/\*----------------------------------------------------------------------------\*/

void colocarDatoPuntajesPorFecha(ListaPuntajesPorFecha &lista, PuntajesPorFecha &puntajes, PtrNodoListaPuntajesPorFecha ptrNodo) {

if ( (! listaVaciaPuntajesPorFecha(lista)) && (ptrNodo != finPuntajesPorFecha()))

ptrNodo->dato = puntajes;

}

/\*----------------------------------------------------------------------------\*/

void obtenerDatoPuntajesPorFecha(ListaPuntajesPorFecha &lista, PuntajesPorFecha &puntajes, PtrNodoListaPuntajesPorFecha ptrNodo) {

if ((! listaVaciaPuntajesPorFecha(lista)) && (ptrNodo != finPuntajesPorFecha()))

puntajes = ptrNodo->dato;

}

/\*----------------------------------------------------------------------------\*/

void eliminarNodoPuntajesPorFecha(ListaPuntajesPorFecha &lista, PtrNodoListaPuntajesPorFecha ptrNodo) {

PtrNodoListaPuntajesPorFecha ptrPrevio;

/\* verifica que la lista no esté vacia y que nodo no sea finJugador\*/

if ((! listaVaciaPuntajesPorFecha(lista)) && (ptrNodo != finPuntajesPorFecha())) {

if (ptrNodo == primero(lista))

lista.primero = siguiente(lista,primero(lista));

else {

ptrPrevio = anterior( lista , ptrNodo );

ptrPrevio->sgte = ptrNodo->sgte;

}

// Si el dato es un TDA, acá habría que llamar al destructor.

delete ptrNodo;

}

}

/\*----------------------------------------------------------------------------\*/

void eliminarNodoPrimeroPuntajesPorFecha(ListaPuntajesPorFecha &lista) {

if (! listaVaciaPuntajesPorFecha(lista))

eliminarNodoPuntajesPorFecha(lista,primero(lista));

}

/\*----------------------------------------------------------------------------\*/

void eliminarNodoUltimoPuntajesPorFecha(ListaPuntajesPorFecha &lista) {

if (! listaVaciaPuntajesPorFecha(lista))

eliminarNodoPuntajesPorFecha(lista,ultimo(lista));

}

/\*----------------------------------------------------------------------------\*/

void eliminarListaPuntajesPorFecha(ListaPuntajesPorFecha &lista) {

/\* retira uno a uno los nodos de la lista \*/

while (! listaVaciaPuntajesPorFecha(lista))

eliminarNodoPuntajesPorFecha(lista,primero(lista));

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha localizarDatoPuntajes(ListaPuntajesPorFecha &lista, PuntajesPorFecha puntajes) {

bool encontrado = false;

PuntajesPorFecha datoCursor;

PtrNodoListaPuntajesPorFecha ptrCursor = primero(lista);

/\* recorre los nodos hasta llegar al último o hasta

encontrar el nodo buscado \*/

while ((ptrCursor != finPuntajesPorFecha()) && (! encontrado)) {

/\* obtiene el dato del nodo y lo compara \*/

obtenerDatoPuntajesPorFecha(lista,datoCursor,ptrCursor);

if (compararDatoPuntajesPorFecha(datoCursor,puntajes) == IGUALPUNTAJEPORFECHA)

encontrado = true;

else

ptrCursor = siguiente(lista,ptrCursor);

}

/\* si no lo encontró devuelve finJugador \*/

if (! encontrado)

ptrCursor = finPuntajesPorFecha();

return ptrCursor;

}

/\*----------------------------------------------------------------------------\*/

void eliminarDatoPuntajesPorFecha(ListaPuntajesPorFecha &lista, PuntajesPorFecha puntajes) {

/\* localiza el dato y luego lo elimina \*/

PtrNodoListaPuntajesPorFecha ptrNodo = localizarDatoPuntajes(lista,puntajes);

if (ptrNodo != finPuntajesPorFecha())

eliminarNodoPuntajesPorFecha(lista,ptrNodo);

}

/\*----------------------------------------------------------------------------\*/

PtrNodoListaPuntajesPorFecha insertarDato(ListaPuntajesPorFecha &lista, PuntajesPorFecha puntajes) {

PtrNodoListaPuntajesPorFecha ptrPrevio = primero(lista);

PtrNodoListaPuntajesPorFecha ptrCursor = primero(lista);

PtrNodoListaPuntajesPorFecha ptrNuevoNodo;

PuntajesPorFecha datoCursor;

bool ubicado = false;

/\* recorre la lista buscando el lugar de la inserción \*/

while ((ptrCursor != finPuntajesPorFecha()) && (! ubicado)) {

obtenerDatoPuntajesPorFecha(lista,datoCursor,ptrCursor);

if (compararDatoPuntajesPorFecha(datoCursor,puntajes) == MAYORPUNTAJEPORFECHA)

ubicado = true;

else {

ptrPrevio = ptrCursor;

ptrCursor = siguiente(lista,ptrCursor);

}

}

if (ptrCursor == primero(lista))

ptrNuevoNodo = adicionarPrincipio(lista,puntajes);

else

ptrNuevoNodo = adicionarDespues(lista,puntajes,ptrPrevio);

return ptrNuevoNodo;

}

/\*----------------------------------------------------------------------------\*/

void reordenarPuntajesPorFecha(ListaPuntajesPorFecha &lista) {

ListaPuntajesPorFecha temp = lista;

PtrNodoListaPuntajesPorFecha ptrCursor = primero(temp);

crearListaPuntajesPorFecha(lista);

while ( ptrCursor != finPuntajesPorFecha() ) {

PuntajesPorFecha puntajes;

obtenerDatoPuntajesPorFecha( temp, puntajes, ptrCursor);

insertarDato( lista, puntajes );

eliminarNodoPuntajesPorFecha( temp, ptrCursor );

ptrCursor = primero(temp);

}

eliminarListaPuntajesPorFecha( temp );

}

/\*----------------------------------------------------------------------------\*/

int longitudPuntajesPorFecha(ListaPuntajesPorFecha &lista){

PtrNodoListaPuntajesPorFecha ptrCursor = primero(lista);

int longitud = 0;

while ( ptrCursor != finPuntajesPorFecha() ) {

longitud++;

ptrCursor = siguiente( lista, ptrCursor);

}

return longitud;

}

/\*----------------------------------------------------------------------------\*/