/\* TDA Lista

\* Implementación Simplemente Enlazada

\* Archivo : ListaPuntajesTotal.cpp

\* Versión : 1.1

\*/

#include "ListaPuntajesTotal.h"

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

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

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

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

/\* Funciones Adicionales \*/

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

/\*

pre : ninguna.

post: compara ambos dato1 y dato2, devuelve

mayorpuntajetotal si dato1 es mayor que dato2,

igualpuntajetotal si dato1 es igual a dato2,

menorpuntajetotal si dato1 es menor que dato2.

dato1.puntaje : dato a comparar.

dato2.puntaje : dato a comparar.

return resultado de comparar dato1 respecto de dato2.

\*/

ResultadoComparacionPuntajesTotal compararDatoPuntajesTotal(PuntajesTotal dato1, PuntajesTotal dato2) {

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

return MAYORPUNTAJETOTAL;

}

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

return MENORPUNTAJETOTAL;

}

else {

return IGUALPUNTAJETOTAL;

}

}

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

/\* Implementación de Primitivas \*/

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

void crearListaPuntajesTotal(ListaPuntajesTotal &lista) {

lista.primero = finPuntajesTotal();

}

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

bool listaVaciaPuntajesTotal(ListaPuntajesTotal &lista) {

return (primero(lista) == finPuntajesTotal());

}

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

PtrNodoListaPuntajesTotal finPuntajesTotal() {

return NULL;

}

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

PtrNodoListaPuntajesTotal primero(ListaPuntajesTotal &lista) {

return lista.primero;

}

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

PtrNodoListaPuntajesTotal siguiente(ListaPuntajesTotal &lista, PtrNodoListaPuntajesTotal ptrNodo) {

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

if ((! listaVaciaPuntajesTotal(lista)) && (ptrNodo != finPuntajesTotal()))

return ptrNodo->sgte;

else

return finPuntajesTotal();

}

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

PtrNodoListaPuntajesTotal anterior(ListaPuntajesTotal &lista, PtrNodoListaPuntajesTotal ptrNodo) {

PtrNodoListaPuntajesTotal ptrPrevio = finPuntajesTotal();

PtrNodoListaPuntajesTotal ptrCursor = primero(lista);

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

ptrPrevio = ptrCursor;

ptrCursor = siguiente(lista,ptrCursor);

}

return ptrPrevio;

}

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

PtrNodoListaPuntajesTotal ultimo(ListaPuntajesTotal &lista) {

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

return anterior(lista,finPuntajesTotal());

}

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

PtrNodoListaPuntajesTotal crearNodoLista(PuntajesTotal dato) {

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

PtrNodoListaPuntajesTotal ptrAux = new NodoListaPuntajesTotal();

ptrAux->dato = dato;

ptrAux->sgte = finPuntajesTotal();

return ptrAux;

}

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

PtrNodoListaPuntajesTotal adicionarPrincipio(ListaPuntajesTotal &lista, PuntajesTotal puntajes) {

/\* crea el nodo \*/

PtrNodoListaPuntajesTotal ptrNuevoNodo = crearNodoLista(puntajes);

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

ptrNuevoNodo->sgte = lista.primero;

lista.primero = ptrNuevoNodo;

return ptrNuevoNodo;

}

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

PtrNodoListaPuntajesTotal adicionarDespues(ListaPuntajesTotal &lista, PuntajesTotal puntajes, PtrNodoListaPuntajesTotal ptrNodo) {

PtrNodoListaPuntajesTotal ptrNuevoNodo = finPuntajesTotal();

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

if (listaVaciaPuntajesTotal(lista))

ptrNuevoNodo = adicionarPrincipio(lista,puntajes);

else {

if (ptrNodo != finPuntajesTotal()) {

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

ptrNuevoNodo = crearNodoLista(puntajes);

ptrNuevoNodo->sgte = ptrNodo->sgte;

ptrNodo->sgte = ptrNuevoNodo;

}

}

return ptrNuevoNodo;

}

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

PtrNodoListaPuntajesTotal adicionarFinal(ListaPuntajesTotal &lista, PuntajesTotal puntajes) {

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

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

}

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

PtrNodoListaPuntajesTotal adicionarAntes(ListaPuntajesTotal &lista, PuntajesTotal puntajes, PtrNodoListaPuntajesTotal ptrNodo) {

PtrNodoListaPuntajesTotal ptrNuevoNodo = finPuntajesTotal();

if (! listaVaciaPuntajesTotal(lista)) {

if (ptrNodo != primero(lista))

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

else

ptrNuevoNodo = adicionarPrincipio(lista,puntajes);

}

return ptrNuevoNodo;

}

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

void colocarDatoPuntajesTotal(ListaPuntajesTotal &lista, PuntajesTotal &puntajes, PtrNodoListaPuntajesTotal ptrNodo) {

if ( (! listaVaciaPuntajesTotal(lista)) && (ptrNodo != finPuntajesTotal()))

ptrNodo->dato = puntajes;

}

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

void obtenerDatoPuntajesTotal(ListaPuntajesTotal &lista, PuntajesTotal &puntajes, PtrNodoListaPuntajesTotal ptrNodo) {

if ((! listaVaciaPuntajesTotal(lista)) && (ptrNodo != finPuntajesTotal()))

puntajes = ptrNodo->dato;

}

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

void eliminarNodoPuntajesTotal(ListaPuntajesTotal &lista, PtrNodoListaPuntajesTotal ptrNodo) {

PtrNodoListaPuntajesTotal ptrPrevio;

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

if ((! listaVaciaPuntajesTotal(lista)) && (ptrNodo != finPuntajesTotal())) {

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 eliminarNodoPrimeroPuntajesTotal(ListaPuntajesTotal &lista) {

if (! listaVaciaPuntajesTotal(lista))

eliminarNodoPuntajesTotal(lista,primero(lista));

}

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

void eliminarNodoUltimoPuntajesTotal(ListaPuntajesTotal &lista) {

if (! listaVaciaPuntajesTotal(lista))

eliminarNodoPuntajesTotal(lista,ultimo(lista));

}

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

void eliminarListaPuntajesTotal(ListaPuntajesTotal &lista) {

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

while (! listaVaciaPuntajesTotal(lista))

eliminarNodoPuntajesTotal(lista,primero(lista));

}

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

PtrNodoListaPuntajesTotal localizarDatoPuntajes(ListaPuntajesTotal &lista, PuntajesTotal puntajes) {

bool encontrado = false;

PuntajesTotal datoCursor;

PtrNodoListaPuntajesTotal ptrCursor = primero(lista);

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

encontrar el nodo buscado \*/

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

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

obtenerDatoPuntajesTotal(lista,datoCursor,ptrCursor);

if (compararDatoPuntajesTotal(datoCursor,puntajes) == IGUALPUNTAJETOTAL)

encontrado = true;

else

ptrCursor = siguiente(lista,ptrCursor);

}

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

if (! encontrado)

ptrCursor = finPuntajesTotal();

return ptrCursor;

}

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

void eliminarDatoPuntajesTotal(ListaPuntajesTotal &lista, PuntajesTotal puntajes) {

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

PtrNodoListaPuntajesTotal ptrNodo = localizarDatoPuntajes(lista,puntajes);

if (ptrNodo != finPuntajesTotal())

eliminarNodoPuntajesTotal(lista,ptrNodo);

}

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

PtrNodoListaPuntajesTotal insertarDato(ListaPuntajesTotal &lista, PuntajesTotal puntajes) {

PtrNodoListaPuntajesTotal ptrPrevio = primero(lista);

PtrNodoListaPuntajesTotal ptrCursor = primero(lista);

PtrNodoListaPuntajesTotal ptrNuevoNodo;

PuntajesTotal datoCursor;

bool ubicado = false;

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

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

obtenerDatoPuntajesTotal(lista,datoCursor,ptrCursor);

if (compararDatoPuntajesTotal(datoCursor,puntajes) == MAYORPUNTAJETOTAL)

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 reordenarPuntajesTotal(ListaPuntajesTotal &lista) {

ListaPuntajesTotal temp = lista;

PtrNodoListaPuntajesTotal ptrCursor = primero(temp);

crearListaPuntajesTotal(lista);

while ( ptrCursor != finPuntajesTotal() ) {

PuntajesTotal puntajes;

obtenerDatoPuntajesTotal( temp, puntajes, ptrCursor);

insertarDato( lista, puntajes );

eliminarNodoPuntajesTotal( temp, ptrCursor );

ptrCursor = primero(temp);

}

eliminarListaPuntajesTotal( temp );

}

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

int longitudPuntajesTotal(ListaPuntajesTotal &lista){

PtrNodoListaPuntajesTotal ptrCursor = primero(lista);

int longitud = 0;

while ( ptrCursor != finPuntajesTotal() ) {

longitud++;

ptrCursor = siguiente( lista, ptrCursor);

}

return longitud;

}

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