#include "listacliente.h"

#include <iostream>

**using namespace std**;

**void** ListaCliente::intercambiar(NodoCliente\*a, NodoCliente\*b) {

}

**void** ListaCliente::copiarTodo(**const** ListaCliente&l) {

}

ListaCliente::ListaCliente() : ultimo(**nullptr**), auxiliar(**nullptr**), auxiliar2(**nullptr**), primerInsertado(**nullptr**) {

}

ListaCliente::ListaCliente(**const** ListaCliente&l) {

}

ListaCliente::~ListaCliente() {

}

**bool** ListaCliente::isEmpty() {

**return** ultimo == **nullptr**;

}

**void** ListaCliente::insertar(**const** Cliente&c) {

NodoCliente\* nuevo\_nodo = **new** NodoCliente(c);

**if** (primerInsertado == **nullptr**) {

primerInsertado = nuevo\_nodo;

}

nuevo\_nodo->setSiguiente(ultimo);

ultimo = nuevo\_nodo;

}

**void** ListaCliente::eleminar(**const** Cliente&c) {

**if**(isEmpty()) {

**throw** ListException("La lista esta vacia,ListaCliente->eliminar");

}

auxiliar = ultimo;

**if**(ultimo->getCliente()==c) {

ultimo=ultimo->getSiguiente();

**delete** auxiliar;

} **else** {

auxiliar = auxiliar->getSiguiente();

**while**(auxiliar != **nullptr**) {

**if**(auxiliar->getCliente() == c) {

auxiliar2 = anterior(auxiliar);

auxiliar2->setSiguiente(auxiliar->getSiguiente());

**delete** auxiliar;

}

auxiliar=auxiliar->getSiguiente();

}

}

}

NodoCliente\* ListaCliente::primerNodo() {

}

NodoCliente\* ListaCliente::ultimoNodo() {

;

}

NodoCliente\* ListaCliente::siguiente(NodoCliente\*c) {

}

NodoCliente\* ListaCliente::anterior(NodoCliente\*c) {

**if**(ultimo == c) {

**throw** ListException("No hay anterior del ultimo insertado, ListaCliente->anterior");

}

auxiliar = ultimo;

**while**(auxiliar != **nullptr**) {

**if**(auxiliar->getSiguiente() == c) {

**return** auxiliar;

}

auxiliar= auxiliar->getSiguiente();

}

}

NodoCliente\* ListaCliente::localiza(**const** Cliente&c) {

**if**(isEmpty() == **true**) {

**throw** ListException("La lista esta vacia,ListaCliente->localiza");

}

auxiliar = ultimo;

**while**(auxiliar != **nullptr**) {

**if**(auxiliar->getCliente() == c) {

**return** auxiliar;

}

auxiliar= auxiliar->getSiguiente();

}

**if**(auxiliar == **nullptr**) {

**throw** ListException("No encontrado,ListaCliente->localiza");

}

}

**void** ListaCliente::ordena() {

}

**void** ListaCliente::ordena(NodoCliente\*leftedge, NodoCliente\*rightedge) {

}

string ListaCliente::recupera(const Cliente&c) {

}

void ListaCliente::guardarAlDisco(const string& fileName) {

}

void ListaCliente::leerDelDisco(const string& fileName) {

}

void ListaCliente::eliminarTodo() {

if(isEmpty()) {

throw ListException("La lista esta vacia,ListaCliente->eliminarTodo");

}

auxiliar = ultimo;

while(auxiliar != nullptr) {

auxiliar2 = auxiliar;

auxiliar = auxiliar->getSiguiente();

delete auxiliar2;

}

ultimo = nullptr;

primerInsertado = nullptr;

}

ListaCliente& ListaCliente::operator=(const ListaCliente&l) {

if(isEmpty()==false) {

eliminarTodo();

}

copiarTodo(l);

return \*this;

}