#include "cliente.h"

**using namespace std**;

Cliente::Cliente() { }

Cliente::Cliente(**const** Cliente&c) : nombre(c.nombre), horaDeLlamada(c.horaDeLlamada), duracion(c.duracion) { }

**string** Cliente::getNombre() {

**return** nombre.toString();

}

**string** Cliente::gethoraDeLlamada() {

**return** horaDeLlamada.toString();

}

**string** Cliente::getDuracion() {

**return** duracion.toString();

}

**string** Cliente::toString() {

**string** resultado;

resultado = "Hora de llamada: ";

resultado += horaDeLlamada.toString();

resultado += " | ";

resultado += "Duracion: ";

resultado += duracion.toString();

resultado += " | ";

resultado += "Nombre: ";

resultado += nombre.toString();

**return** resultado;

}

**void** Cliente::setNombre(**const** Nombre&n) {

nombre = n;

}

**void** Cliente::setHoraDeLlamada(**const** Horario&h) {

horaDeLlamada = h;

}

**void** Cliente::setDuracion(**const** Duracion&d) {

duracion = d;

}

Cliente& Cliente::**operator**=(**const** Cliente&c) {

nombre = c.nombre;

horaDeLlamada = c.horaDeLlamada;

duracion = c.duracion;

**return** \***this**;

}

**bool** Cliente::**operator**==(**const** Cliente&c) {

**if**( nombre == c.nombre and horaDeLlamada == c.horaDeLlamada){

**return true**;

}

**return false**;

}

**bool** Cliente::**operator**<(**const** Cliente&c) {

**return** horaDeLlamada < c.horaDeLlamada;

}

**bool** Cliente::**operator**<=(**const** Cliente&c) {

**return** horaDeLlamada <= c.horaDeLlamada;

}

**bool** Cliente::**operator**>(**const** Cliente&c) {

**return** horaDeLlamada > c.horaDeLlamada;

}

**bool** Cliente::**operator**>=(**const** Cliente&c) {

**return** horaDeLlamada >= c.horaDeLlamada;

}

ostream& **operator**<<(ostream& os,Cliente& c){

}

**istream**& **operator**>>(**istream**& is,Cliente& c){

}

#include "nodocliente.h"

**using namespace std**;

NodoCliente::NodoCliente() : siguiente(**nullptr**) { }

NodoCliente::NodoCliente(**const** Cliente&c)

{

cliente = c;

}

Cliente& NodoCliente::getCliente() {

**return** cliente;

}

NodoCliente\* NodoCliente::getSiguiente() {

**return** siguiente;

}

**string** NodoCliente::toString() {

**return** cliente.toString();

}

**void** NodoCliente::setCliente(**const** Cliente&c) {

cliente = c;

}

**void** NodoCliente::setSiguiente(NodoCliente\*s) {

siguiente = s;

}

#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) {

}

NodoCliente\* ListaCliente::primerNodo() {

}

NodoCliente\* ListaCliente::ultimoNodo() {

}

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

}

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

}

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

}

**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() {

}