“main.cpp”

```
#include "iostream"
#include "medicos.h"
#include "citas.h"
using namespace std;

void procesarPeticiones(const tListaCitas& listaCitas, tListaMedicos& listaMedicos);

int main()
{
	bool ok1, ok2;
	tListaMedicos listaMedicos;
	tListaCitas listaCitas;

	_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);
	inicializarListaMedicos(listaMedicos);
	inicializarListaCitas(listaCitas);
	ok1 = leerMedicos(listaMedicos);
	ok2 = leerCitas(listaCitas);

	if (ok1 && ok2)
	{
		mostrarMedicos(listaMedicos);
		mostrarPeticiones(listaCitas);
		procesarPeticiones(listaCitas,listaMedicos);
		mostrarMedicos(listaMedicos);
		destruir(listaCitas);
		destruir(listaMedicos);
	}
	return 0;
}
```

```
void procesarPeticiones(const tListaCitas& listaCitas, tListaMedicos& listaMedicos)
{
	int codigo_medico, libres;
	tMedicoPtr ptr_medico;
	float total = 0;
	int num_citas = getNumElems(listaCitas);

	for (int i = 0; i < num_citas; i++)
	{
		codigo_medico = getCodigoMedico(listaCitas, i);
		ptr_medico = buscar(listaMedicos, codigo_medico);
		if (ptr_medico != nullptr)
		{
			total += getTarifaMedico(ptr_medico);
			libres = getLibresMedico(ptr_medico);
			setLibresMedico(ptr_medico, libres - 1);
			if (libres-1 == 0)
				eliminar(listaMedicos, ptr_medico);
		}
	}
	cout << "El importe total es :" << total << " euros" << endl << endl;
}
```

“médicos.h”

```
#include <string>
using namespace std;

const int MAX_MEDICOS = 100;

typedef struct
{
	int codigo;
	int libres;
	float tarifa;
} tMedico;

typedef tMedico* tMedicoPtr;
typedef tMedicoPtr tArray[MAX_MEDICOS];

typedef struct
{
	tArray medicos;
	int contador;
} tListaMedicos;

void inicializarListaMedicos(tListaMedicos& listaMedicos);
bool leerMedicos(tListaMedicos &lista);
void mostrarMedicos(const tListaMedicos& listaMedicos);

tMedicoPtr buscar(const tListaMedicos &listaMedicos, int codigo);
int getTarifaMedico(tMedicoPtr ptr_medico);
int getLibresMedico(tMedicoPtr ptr_medico);
void setLibresMedico(tMedicoPtr ptr_medico, int libres);

void eliminar(tListaMedicos& listaMedicos, tMedicoPtr ptr_medico);
void destruir(tListaMedicos& listaMedicos);
```

“médicos.cpp”

```
#include <fstream>
#include <iostream>
#include "medicos.h"
using namespace std;

int buscarPosMedico(const tListaMedicos& listaMedicos, tMedicoPtr ptr_medico);

void inicializarListaMedicos(tListaMedicos& listaMedicos)
{
	listaMedicos.contador = 0;
}

bool leerMedicos(tListaMedicos& lista)
{
	bool cargado = true;
	ifstream entrada;
	tMedico aux;
	int codigo;

	entrada.open("medicos.txt");

	if (entrada.is_open())
	{
		entrada >> codigo;
		while (codigo != -1 && lista.contador < MAX_MEDICOS)
		{
			aux.codigo = codigo;
			entrada >> aux.libres >> aux.tarifa;
			lista.medicos[lista.contador] = new tMedico(aux);
			lista.contador++;
			entrada >> codigo;
		}
		entrada.close();
	}
	else
	{
		cout << "Error, no se pudo abrir el archivo 'Medicos.txt'" << endl;
		cargado = false;
	}
	return cargado;
}
```

```
tMedicoPtr buscar(const tListaMedicos& listaMedicos, int codigo)
{
	tMedicoPtr ptr_medico = nullptr;
	bool encontrado = false;
	int inicio = 0;
	int fin = listaMedicos.contador - 1;
	int mitad;

	while (inicio <= fin && !encontrado)
	{
		mitad = (inicio + fin) / 2;
		if (listaMedicos.medicos[mitad]->codigo == codigo)
		{
			ptr_medico = listaMedicos.medicos[mitad];
			encontrado = true;
		}
		else if (codigo > listaMedicos.medicos[mitad]->codigo)
			fin = mitad - 1;
		else
			inicio = mitad + 1;
	}
	return ptr_medico;
}

int getTarifaMedico(tMedicoPtr ptr_medico)
{
	return ptr_medico->tarifa;
}

int getLibresMedico(tMedicoPtr ptr_medico)
{
	return ptr_medico->libres;
}

void setLibresMedico(tMedicoPtr ptr_medico, int libres)
{
	ptr_medico->libres = libres;
}
```

```
void mostrarMedicos(const tListaMedicos& listaMedicos) {
	cout << "   *****   LISTA DE MÉDICOS   *****" << endl <<endl;
	cout << "======================================" << endl;
	for (int i = 0; i < listaMedicos.contador; i++) {
		cout << listaMedicos.medicos[i]->codigo << " "
			<< listaMedicos.medicos[i]->libres << " "
			<< listaMedicos.medicos[i]->tarifa << endl;
	}
	cout << "======================================" << endl <<endl;
}

void eliminar(tListaMedicos& listaMedicos, tMedicoPtr ptr_medico)
{

	int pos = buscarPosMedico(listaMedicos, ptr_medico);
	delete ptr_medico;
	for (int i = pos + 1; i < listaMedicos.contador; i++)
		listaMedicos.medicos[i - 1] = listaMedicos.medicos[i];
	listaMedicos.contador--;
}

int buscarPosMedico(const tListaMedicos& listaMedicos, tMedicoPtr ptr_medico)
{
	int i = 0;
	while (i < listaMedicos.contador && listaMedicos.medicos[i] != ptr_medico)
		i++;
	return i;
}
void destruir(tListaMedicos& listaMedicos)
{
	for (int i = 0; i < listaMedicos.contador; i++) {
		delete listaMedicos.medicos[i];
	}
	listaMedicos.contador = 0;
}
```

“citas.h”

```
#pragma once
#include <string>
using namespace std;

typedef struct
{
	int codigoMedico;
	string paciente;
} tCita;

typedef struct
{
	tCita* citas;
	int contador, capacidad;
}tListaCitas;

void inicializarListaCitas(tListaCitas& listaCitas);
bool leerCitas(tListaCitas& lista);
void mostrarPeticiones(const tListaCitas& listaCitas);
void ampliar(tListaCitas& lista);
void destruir(tListaCitas& listaCitas);
int getNumElems(const tListaCitas& listaCitas);
int getCodigoMedico(const tListaCitas& listaCitas, int pos);
```

“citas.cpp”

```
#include "citas.h"
#include <fstream>
#include <iostream>
using namespace std;

void inicializarListaCitas(tListaCitas& listaCitas)
{
	listaCitas.contador = 0;
	listaCitas.capacidad = 20;
	listaCitas.citas = new tCita[listaCitas.capacidad];

}bool leerCitas(tListaCitas& lista) {
	bool cargado = true;
	ifstream entrada;
	int codigo;
	char espacio;

	entrada.open("Citas.txt");
	if (entrada.is_open())
	{
		entrada >> codigo;
		while (codigo != -1)
		{
			if (lista.contador == lista.capacidad)
				ampliar(lista);

			lista.citas[lista.contador].codigoMedico = codigo;
			entrada.get(espacio);
			getline(entrada, lista.citas[lista.contador].paciente);
			lista.contador++;
			entrada >> codigo;
		}
		entrada.close();
	}
	else {
		cout << "Error, no se pudo abrir el archivo 'Citas.txt'" << endl;
		cargado = false;
	}

	return cargado;
}
```

```
void mostrarPeticiones(const tListaCitas& listaCitas) {

	cout << "   *****   LISTA DE PETICIONES DE CITAS PENDIENTES DE PROCESAR
*****" << endl << endl;

	for (int i = 0; i < listaCitas.contador; i++) {
		cout << listaCitas.citas[i].codigoMedico << " - "
			<< listaCitas.citas[i].paciente << endl;
	}
	cout << endl;
}

void ampliar(tListaCitas& lista)
{
	tCita* aux = new tCita[2 * lista.capacidad];
	for (int i = 0; i < lista.contador; i++)
		aux[i] = lista.citas[i];
	delete[] lista.citas;
	lista.citas = aux;
	lista.capacidad *= 2;
	aux = nullptr;
}

void destruir(tListaCitas& listaCitas)
{
	delete[] listaCitas.citas;
	listaCitas.citas = nullptr;
	listaCitas.contador = 0;
}

int getNumElems(const tListaCitas& listaCitas)
{
	return listaCitas.contador;
}

int getCodigoMedico(const tListaCitas& listaCitas, int pos)
{
	return listaCitas.citas[pos].codigoMedico;
}
```