Ejercicio de Laboratorio (11/02/2025)

Fundamentos de la Programación II

Grupo E

Parte I

En este ejercicio debes implementar el mismo ejercicio del 04/02/2025, pero ahora usando módulos para implementar el tipo `tMatriz`. Para ello agrega los ficheros `matrices.h` y `matrices.cpp` al proyecto. No olvides que `matrices.cpp` debe contener `#include "matrices.h"`. El fichero `matrices.h` es el siguiente:

```
#include <fstream>
#include <iomanip>

using namespace std;

const int MAX = 10;

typedef struct {
  int dimension;
  int matriz[MAX][MAX];
} tMatriz;

void inicia(tMatriz& m, int d);
bool posicionValida(const tMatriz& m, int i, int j);
int getDimension(const tMatriz& m);
int getElem(const tMatriz& m, int i, int j);
void setElem(tMatriz& m, int i, int j, int n);
```

Las funciones que debes implementar en `matrices.cpp` deben hacer lo siguiente:

- `inicia(m,d)`: inicia la dimensión de la matriz `m` al valor `d`.
- `posicionValida(m,i,j)`: devuelve `true` si y solo si `(i,j)` está dentro del rango de la matriz `m`.
- `getDimension(m)`: devuelve la dimensión de la matriz `m`, es decir, el valor del campo `dimension`.
- `getElem(m,i,j)`: devuelve el elemento `(i,j)` de la matriz `m`.
- `setElem(m,i,j,n)`: coloca en la posición `(i,j)` de la matriz `m`, el elemento `n`.

Crea otro modulo `operadores`, donde implementar los operadores "<<" y ">>" para el tipo `tMatriz`. Concretamente `operadores.h` tendrá la siguiente forma:

```
#pragma once
#include <iostream>
#include <fstream>
#include "matrices.h"

using namespace std;

ostream& operator<<(ostream& out, const tMatriz& m);
ifstream& operator>>(ifstream& archivo, tMatriz& m);
```

Implementa estos operadores en `operadores.cpp`.

En `main.cpp` deben estar las implementaciones de las funciones `cargaMatriz`, `esPuntoSumidero` , `muestraPuntosSumidero` e `intercambiaDiagonales`, que no deben acceder al tipo `tMatriz`, sino que deben usar las funciones públicas que ofrece el módulo `matrices.h`. La función `main` debe hacer lo mismo que en el ejercicio anterior, pero adaptando la sintaxis, si es necesario, al uso de módulos.

Parte II

Modifica el ejercicio de laboratorio del día 28/01/2025, para incluir módulos. Declara e implementa el tipo `tCiudad` en un módulo. Implementa también el tipo `tListaCiudades` en otro módulo. Concretamente, el `.h` para la implementación del tipo `tCiudad` será:

```
#pragma once
#include <fstream>
#include <iomanip>

using namespace std;

const int MAX_FILAS = 20;
const int MAX_COLUMNAS = 20;

typedef struct {
  int nFilas;
  int nCols;
  int matriz[MAX_FILAS][MAX_COLUMNAS];
} tCiudad;

void inicia(tCiudad& c, int nf,int nc);
bool posicionValida(const tCiudad& c, int i, int j);
int getNumFilas(const tCiudad& c);
int getNumColumnas(const tCiudad& c);
int getElem(const tCiudad& c, int i, int j);
void setElem(tCiudad& c, int i, int j, int n);
```

donde las funciones hacen lo siguiente:

- `inicia(m,nf,nc)`: pone el número de filas y columnas de la matriz `m` a los valores especificados en los parámetros.

- `posicionValida(m,i,j)`: devuelve `true` si y solo si `(i,j)` está dentro del rango de la matriz `m`.
- `getNumFilas(m)`: devuelve el número de filas de la matriz `m`.
- `getNumColumnas(m)`: devuelve el número de columnas de la matriz `m`.
- `getElem(m,i,j)`: devuelve el elemento `(i,j)` de la matriz `m`.
- `setElem(m,i,j,n)`: coloca el valor `n` en la posición `(i,j)` de la matriz `m`.

El tipo `tListaCiudades` lo implementaremos en otro módulo. En este caso el `.h` del módulo será el siguiente:

```
#pragma once
#include "matrices.h"

const int MAX_CIUDADES = 30;

typedef struct {
  int cont;
  tCiudad listaC[MAX_CIUDADES];
} tListaCiudades;

void inicia(tListaCiudades& lc);
int dameLongitud(const tListaCiudades& lc);
tCiudad getElem(const tListaCiudades& lc, int i);
void insertarAlFinal(tListaCiudades& lc, const tCiudad& m);
```

donde las funciones públicas deben hacer lo siguiente:

- `inicia(lc)`: pone la longitud de la lista `lc` a `0`.
- `dameLongitud(lc)`: devuelve la longitud de la lista `lc`.
- `getElem(lc,i)`: devuelve el elemento i-ésimo de la lista `lc`.
- `insertarAlFinal(lc,c)`: inserta `c` al final de la lista `lc`.

El fichero `main.cpp` permanece igual salvo los cambios de sintaxis para adaptarlo al uso de módulos. Al igual que en la parte I, crea un módulo `operadores`, donde `operadores.h` será:

```
#pragma once

#include "ListaCiudades.h"

ifstream& operator>>(ifstream& archivo, tCiudad& c);
ostream& operator<<(ostream& out, const tCiudad& c);
ifstream& operator>>(ifstream& archivo, tListaCiudades& lc);
ostream& operator<<(ostream& out, const tListaCiudades& lc);
```