Facultad de Informática – Universidad Complutense de Madrid

Fundamentos de la programación II

Curso 2024–2025

Ejercicio de Laboratorio: 28 / 01 / 2025

PARTE I

Queremos representar la altura de los edificios de una ciudad. Para ello usaremos el tipo definido `tCiudad`, para almacenar **como máximo** `MAX_FILAS x MAX_COLUMNAS` edificios. Una posición `[i][j]` de la matriz, representa la altura del edificio (`>0`) construido en esa posición. La información de una ciudad se encuentra almacenada en un archivo de texto de nombre "`ciudad.txt`", con el siguiente formato:

```
6 6
8 9  5 4  4  7
2 10 4 9  7  8
1 6  7 5  23 3
6 13 3 11 15 3
3 9  3 7  20 6
2 5  9 13 6  2
```

La primera línea del archivo determina el número de filas (6) y el número de columnas (6) de la ciudad. Las siguientes líneas son las alturas de los edificios que componen la ciudad y que permiten cargarla.

Implementa las constantes y tipos necesarios para representar una ciudad (`tCiudad`). Asume que el máximo de filas y columnas para una ciudad es 50.

Implementa las siguientes funciones:

- ✓ `void cargarCiudad(ifstream& archivo, tCiudad& ciudad)`: que, dado un archivo de texto con el formato anterior, carga la información de una ciudad.

- ✓ `void mostrarCiudad(const tCiudad& ciudad)`: muestra por consola la información de una ciudad. Por ejemplo, para nuestro fichero "`ciudad.txt`", se muestra por consola:

```
8    9    5    4    4    7
2    10   4    9    7    12
1    6    7    5    5    3
6    4    3    11   15   3
3    9    5    7    20   6
2    5    9    13   6    22
```

- ✓ `int edificiosVisiblesDir(const tCiudad& c, int i, int j, int incri, int incrj)`: Devuelve el número de edificios visibles desde la posición `[i][j]` de la ciudad, en la dirección `(incri, incrj)`. **Un edificio es visible desde la posición `i, j` en la dirección `(incri, incrj)`, si no tiene ningún otro edificio delante con una altura superior, sin considerar la posición (i, j).** Por ejemplo, en nuestra ciudad, si consideramos la posición `i = 4, j = 2`, y la dirección `incri = -1, incrj = -1` (diagonal superior izquierda), el número de edificios visibles es 1, que se corresponde con el valor 4. Si tomamos como dirección `incri = -1, incrj = 1` (diagonal superior derecha), el número de edificios visibles es 2, que se corresponde con las alturas 11 y 12. Otro ejemplo sería considerar la posición `i = 3, j = 1` y la dirección `incri = 1, incrj = -1` (diagonal inferior izquierda). En este caso solo hay un edificio visible, que es el `3`.

- ✓ `int edificiosVisibles(const tCiudad& c, int i, int j)`: que devuelve el número de edificios visibles de la ciudad, desde el edificio `[i][j]`. Nuestros edificios únicamente tienen ventanas en las esquinas, es decir, solo podrán ver aquellos edificios que estén en sus diagonales. Para implementar esta función debes utilizar la función anterior, pasándole las direcciones adecuadas. Para almacenar las direcciones de las diagonales y poder recorrerlas, usa una matriz (constante) de la siguiente forma:

```
const int NUM_DIRECCIONES = 4;
const int VECTOR_DIRECCION = 2;

const int DIRECCIONES[NUM_DIRECCIONES][VECTOR_DIRECCION] =
                    { -1, -1, -1, 1, 1, 1, 1, -1 };
```

  Donde los dos primeros elementos `-1, -1` son los valores de las posiciones `[0][0]` y `[0][1]`, es decir son la primera fila de la matriz. Los dos siguientes elementos `-1,1` son las posiciones `[1][0]` y `[1][1]`. Después `1,1` son `[2][0]` y `[2][1]`. Y Finalmente `1, -1` están en `[3][0]` y `[3][1]`. Por ejemplo, si llamamos a esta función con la posición `(4,3)`, devolvería 5 edificios visibles. 1 edificio en la diagonal superior izquierda, 2 en la diagonal superior derecha, 1 en la diagonal inferior izquierda y 1 en la diagonal inferior derecha.

- ✓ Implementa una función main que haga lo siguiente:

  - ✓ Cargue del fichero "`ciudad.txt`" una ciudad.
  - ✓ Si la carga falla, manda un mensaje de error. En otro caso realiza el siguiente bucle: Pide al usuario dos enteros, `fila` y `columna`, comprendidos entre las dimensiones de la ciudad. Si alguno de estos enteros es `-1`, el programa termina. En otro caso muestra el número de edificios visibles desde el edificio situado en la posición `fila, columna` y vuelve a solicitar una `fila` y una `columna`.

A continuación, se muestra un ejemplo de ejecución:

```
   8    9    5    4    4    7
   2   10    4    9    7   12
   1    6    7    5    5    3
   6    4    3   11   15    3
   3    9    5    7   20    6
   2    5    9   13    6   22
Valor de x, entre 0 y 5: 5
Valor de y, entre 0 y 5: 2
Los edificios visibles desde (5,2) = 9 son: 3
Valor de x, entre 0 y 5: 3
Valor de y, entre 0 y 5: 3
Los edificios visibles desde (3,3) = 11 son: 8
Valor de x, entre 0 y 5: 5
Valor de y, entre 0 y 5: 0
Los edificios visibles desde (5,0) = 2 son: 1
Valor de x, entre 0 y 5: 4
Valor de y, entre 0 y 5: 2
Los edificios visibles desde (4,2) = 5 son: 5
Valor de x, entre 0 y 5: -1
Valor de y, entre 0 y 5: 3
```

PARTE II

Añade un nuevo tipo, `tListaCiudades`, para representar una lista de ciudades, de tamaño máximo `10`. Implementa las siguientes funciones:

- `void cargarListaCiudades(ifstream& archivo, tListaCiudades& lc)`: Carga, del fichero "`ciudades.txt`", la lista de ciudades. La primera línea del fichero es un entero que contiene el número de ciudades a cargar. Lógicamente esta función se implementa usando la función `cargarCiudad`.
- `void mostrarListaCiudades(const tListaCiudades& lc)`: muestra la lista de ciudades por consola. Por ejemplo, para el fichero "`ciudades.txt`" mostraría:

```
CIUDAD NUMERO: 0
   8    9    5    4    4    7
   2   10    4    9    7   12
   1    6    7    5    5    3
   6    4    3   11   15    3
   3    9    5    7   20    6
   2    5    9   13    6   22
CIUDAD NUMERO: 1
   3    3    3
   3    3    3
   3    3    3
CIUDAD NUMERO: 2
   1    2    3
   4    5    6
   7    8    9
  10   11   12
CIUDAD NUMERO: 3
   5
   7
```

- `double mediaVisibilidad(const tCiudad& ciudad):` Devuelve la visibilidad media de una ciudad, es decir, la suma del número de edificios visibles desde cada edificio de la ciudad, dividido entre el número de edificios.
- `int mejorCiudad(const tListaCiudades& lc):` Devuelve el índice de la ciudad con mejor visibilidad, es decir, con mejor visibilidad media.
- implementa la función `main` para que cargue la lista de ciudades del fichero correspondiente, las muestre, calcule la ciudad con mejor visibilidad media y posteriormente la muestre.

PARTE III

Implementa operadores en la Parte II, para la lectura y escritura de la lista de ciudades, y de las ciudades. Implementa además el operador "<" de forma que devuelva true si y sólo si la primera ciudad tiene una media de visibilidad mejor que la segunda. Incorpora estos operadores a la Parte II. Concretamente el operador "<" debes usarlo en la función `mejorCiudad`.

Modifica la Parte II para que la búsqueda de edificios visibles no sólo se haga en las diagonales, sino también hacia el norte, sur, este y oeste. Para ello no puedes tocar absolutamente ninguna función. Sólo puedes modificar las constantes `NUM_DIRECCIONES` y `VECTOR_DIRECCION`.