Universidad de Granada
Departamento de Ciencias de la Computación
e Inteligencia Artificial

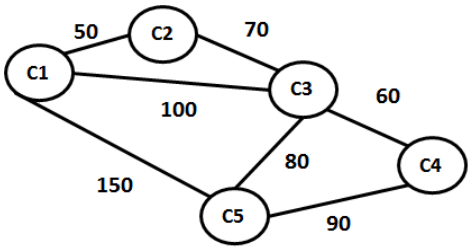
Metodología de la Programación (2016/17)
1º Grado en Ingeniería Informática
Conv. Ordinaria - 14 de Junio de 2017

DECSAI

Normas para la realización del examen: **Duración: 3 horas**

- **El único material permitido durante la realización del examen es un bolígrafo azul o negro.**
- **Debe disponer de un documento oficial que acredite su identidad a disposición del profesor.**
- **No olvide escribir su nombre completo y grupo en todos y cada uno de los folios que entregue.**

Se desea construir la clase `RedCiudades` para almacenar datos de un conjunto de ciudades, y las distancias de los caminos directos que las conectan. Si entre dos ciudades no existe un camino directo, se almacenará un cero. Se supone que la distancia de una ciudad consigo misma será cero y que las distancias son simétricas. Un ejemplo con 5 ciudades sería:



	C1	C2	C3	C4	C5
C1	0	50	100	0	150
C2	50	0	70	0	0
C3	100	70	0	60	80
C4	0	0	60	0	90
C5	150	0	80	90	0

Se propone la siguiente representación para la clase:

```
class RedCiudades {
  private:
    int num_ciudades;     // Número de ciudades
    InfoCiudad * info;    // info[i]: info de la ciudad i
    double ** distancia; // distancia[i][j]: distancia
                         // entre las ciudades i y j
  public:
    // ... interfaz pública de la clase
```

```
struct  InfoCiudad {
   char *  nombre; // Nombre
   int poblacion;  // Num. habs.
};
```

Suponga que dispone de los métodos públicos de consulta:

- `NumCiudades`: devuelve el número de ciudades.
- `Distancia`: devuelve la distancia entre dos ciudades.
- `NombreCiudad`: devuelve el nombre de una ciudad (en realidad, la dirección de inicio).
- `PoblacionCiudad`: devuelve número de habitantes de una ciudad.

Para la realización de los ejercicios propuestos deberá escribir en un folio aparte el fichero `RedCiudades.h`.

Entenderemos que los métodos pedidos en los ejercicios 1, 2, 3 y 4 serán parte de `RedCiudades.cpp` y como tales deberá escribirlos.

◁ Ejercicio 1 ▷ Métodos básicos de la clase [1.25 puntos]

Defina los siguientes métodos para la clase `RedCiudades`:

1. *(0.5 puntos)* Constructor por defecto y destructor. El constructor por defecto crea una *red vacía* (escriba también el método `EstaVacia` que indique si una red está vacía).
2. *(0.75 puntos)* Constructor de copia y operador de asignación.

◁ Ejercicio 2 ▷ Sobrecarga de operadores [1.0 puntos]

Implemente la siguiente funcionalidad para la clase sobrecargando los operadores << y >>:

1. *(0.25 puntos)* operador de salida << para poder insertar en un flujo de salida el contenido de una red en formato texto. En concreto, deberá aparecer:
   - un número entero que indica el número de ciudades y un salto de línea,
   - una serie de líneas (tantas como ciudades) que contienen el índice de la ciudad (un número entero), su nombre (una serie de caracteres sin espacios intermedios) y su población (un número entero),
   - una serie de líneas (tantas como conexiones entre ciudades) que contienen dos números enteros (números de las ciudades conectadas) y un número real (distancia entre ellas). **Nota**: Cada conexión entre ciudades se escribe una sóla vez (no importa cuál es la ciudad de origen o destino).
2. *(0.75 puntos)* operador de entrada >> para poder obtener desde un flujo de entrada el contenido de un objeto. Se asume el mismo formato indicado en el ejercicio anterior.

◁ Ejercicio 3 ▷ Métodos y funciones para E/S [1.25 puntos]

1. *(0.5 puntos)* Constructor con argumento. Recibe el nombre de un fichero de **texto** con la información sobre una red. El fichero debe contener en la primera línea la *cadena mágica* `"RED"` y un salto de línea. A continuación está el contenido de la red en formato de texto, tal como se indica en el ejercicio 2.
2. *(0.5 puntos)* Método `LeerRedCiudades` que permite actualizar el contenido de una red con los datos de un fichero de texto con el formato conocido. El contenido de la red será sustituido por los datos leidos.
3. *(0.25 puntos)* Método `EscribirRedCiudades` que permite guardar el contenido de una red en un fichero de texto, que tendrá el formato descrito en el apartado anterior.

◁ Ejercicio 4 ▷ Métodos de cálculo [1.25 puntos]

1. *(0.5 puntos)* Implemente el método `CiudadMejorConectada` que permita obtener la ciudad (su índice) con mayor número de conexiones directas.
   En el ejemplo la ciudad mejor conectada sería la ciudad C3 con 4 conexiones.
2. *(0.75 puntos)* Implemente el método `MejorEscalaEntre` para que, dadas dos ciudades i y j no conectadas directamente, devuelva aquella ciudad intermedia z que permita hacer el trayecto entre i y j de la forma más económica posible. Es decir, se trata de encontrar una ciudad z tal que $d(i,z) + d(z,j)$ sea mínima ($d(a,b)$ es la distancia entre las ciudades a y b). El método devuelve -1 si no existe dicha ciudad intermedia.
   Por ejemplo, si se desea viajar desde la ciudad C2 a la C5, hacerlo a través de la ciudad C1 tiene un costo de $50 + 150 = 200$ mientras que si se hace a través de la ciudad C3, el costo sería $70 + 80 = 150$.

◁ Ejercicio 5 ▷ Aplicación [1.25 puntos]

Escriba un programa *completo* que reciba desde la línea de órdenes el nombre de un fichero de texto que contiene la descripción de una red (como se indica en el ejercicio 3) y calcule, para todas las parejas de ciudades que *no* estén directamente conectadas, cuál es la mejor escala con una sola ciudad intermedia (su índice). Si no la tuviera deberá indicarlo.