UNIVERSIDAD DE GRANADA

Ciencias de la Computación e Inteligencia Artificial

Metodología de la Programación (2018/19)
1º GII / GII-M / GII-ADE
Conv. Ordinaria - 7 de Junio de 2019

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

- **a) el único material permitido durante la realización del examen es un bolígrafo azul o negro; b) debe disponer de un documento oficial que acredite su identidad a disposición del profesor; c) no olvide escribir su nombre completo y grupo en todos y cada uno de los folios que entregue.**
- **Nota general: deberá implementar los métodos/funciones que necesite si no se indica que están ya disponibles. En el caso en que no se indique de forma explícita la clase en que debe incluir un método, debe decidir cuál es la más adecuada. Tenga también en cuenta que quizás sea necesario implementar un mismo método en varias clases.**

Se desea construir una clase que permita almacenar la información de rutas de vuelo para una compañía aérea. La representación de la trayectoria de vuelo se realiza mediante el almacenamiento de las coordenadas GPS de ciertos puntos de la ruta. Cada punto es un objeto de la clase **Punto3D** y se caracteriza mediante 3 valores de tipo **double** (x: longitud, y: latitud y z: altura). Suponga que dispone de los métodos públicos de **consulta** sobre esta clase: **getX**, **getY** y **getZ**.

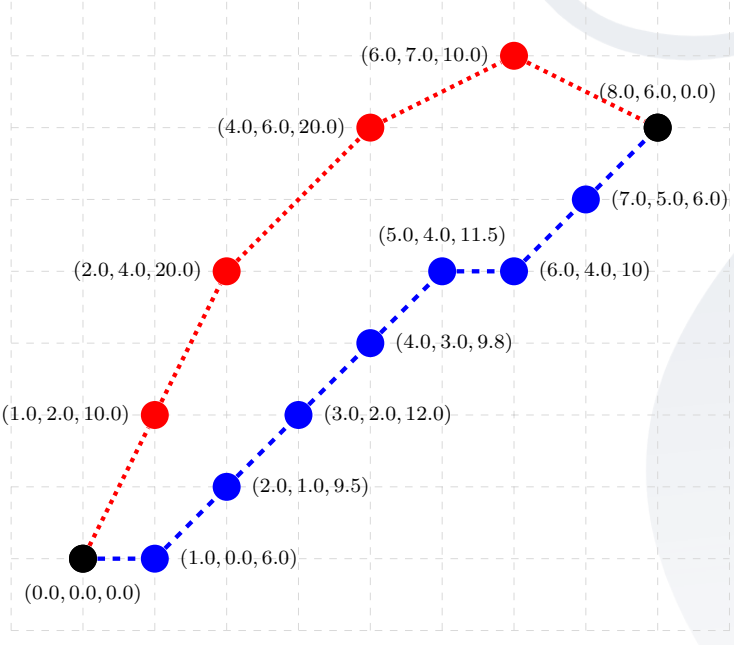
En relación a las rutas se almacenan en realidad dos informaciones diferentes: la primera de ellas consiste en una secuencia de puntos GPS que definen la **ruta óptima** que debería seguir el vuelo (esta información no se modifica una vez que ha sido determinada y contiene un número de puntos que depende de la distancia a cubrir (cada coordenada representa el punto en que debería estar el avión en instantes determinados de tiempo, por ejemplo, cada 10 minutos); la segunda información consiste en la **ruta real** del vuelo. La secuencia de coordenadas reales se va componiendo a medida que trascurre el vuelo y contará con tantos puntos como sea preciso para cubrir la trayectoria real del avión. Se propone la siguiente estructura básica para las clases que permiten modelizar este comportamiento:

```
class Punto3D {
   private:
       double x, y, z;
   public:
       // ... interfaz
};
```

```
class Trayectoria {
   private:
       Punto3D *puntos;
       int numeroPuntos;
   public:
       // ... interfaz
};
```

```
class  Vuelo {
   private:
       int idVuelo;
       Trayectoria rutaOptima;
       Trayectoria rutaReal;
   public:
        // ... interfaz
};
```

La imagen siguiente muestra un ejemplo de vuelo: la línea superior (punteada) representa la ruta óptima y la inferior (a trazos) la real. Para poder representarlo gráficamente se han considerado únicamente las coordenadas **x** e **y**, aunque los valores junto a los puntos indican también el valor de la altura. Se observa que la ruta óptima queda definida por 6 puntos y la real por 9.




```
356
6
0.0 0.0 0.0
1.0 2.0 10.0
2.0 4.0 20.0
4.0 6.0 20.0
6.0 7.0 10.0
8.0 6.0 0.0
9
0.0 0.0 0.0
1.0 0.0 6.0
2.0 1.0 9.5
3.0 2.0 12.0
4.0 3.0 9.8
5.0 4.0 11.5
6.0 4.0 10.0
7.0 5.0 6.0
8.0 6.0 0.0
```

◁ Ejercicio 1 ▷ Métodos básicos de la clase [1.5 puntos]

Defina los siguientes métodos para la clase **Vuelo**:

1. *(0.5 puntos)* **Constructor sin argumentos** (crea un objeto que representa un *vuelo vacío, sin datos*, con **-1** como identificador de vuelo). Indique si es necesario implementar el constructor por defecto de las clases **Trayectoria** y **Punto3D**. En caso afirmativo, haga la implementación. De igual forma, implemente el destructor en las clases en que sea necesario.
2. *(0.5 puntos)* **Constructor de copia** y **operador de asignación**. Implemente estos métodos en las clases en que sea preciso hacerlo.

3. *(0.5 puntos)* **Constructor** de la clase **Vuelo** que recibe como argumentos un identificador de vuelo y la ruta óptima (un objeto de la clase **Trayectoria**). Implemente además un constructor para **Trayectoria** que reciba como argumentos los datos de la ruta: un array de objetos de la clase **Punto3D** y número de puntos del array.

◁ Ejercicio 2 ▷ Sobrecarga de operadores (1) [1 punto]

1. *(0.5 puntos)* Sobrecargue el operador **+=** para la clase **Vuelo**. El objetivo es incorporar un nuevo punto para la ruta real. El nuevo objeto **Punto3D** se añade **al final** del array de puntos de la ruta real. Tenga en cuenta que la primera operación de agregación debe suponer la inclusión de dos puntos: el punto de partida (tomado de la ruta óptima; ocupará la posición 0) y el punto pasado como argumento (en el índice 1). Puede asumirse que en la clase **Trayectoria** se dispone del operador **[]** y del método **getNumeroPuntos**.
2. *(0.5 puntos)* Sobrecargue el operador **<<** (puede asumir que los operadores de salida y entrada están ya disponibles para la clase **Punto3D**) para poder insertar en un flujo de salida el contenido de un vuelo en formato texto. Deberá aparecer tal y como se muestra en la figura incluida al inicio. El formato, como se aprecia, contiene:
   - El número que representa el identificador del vuelo (y salto de línea).
   - Un número entero que indica el número de puntos de la ruta óptima (y salto de línea).
   - Una línea diferente para cada punto de la ruta óptima: para cada punto se indican sus tres coordenadas.
   - Un número entero indicando el número de puntos de la ruta real (y salto de línea).
   - Una línea diferente para las coordenadas de cada punto de la ruta real.

◁ Ejercicio 3 ▷ Métodos y funciones para E/S [0.75 puntos]

Implemente un constructor de la clase **Vuelo** con un argumento que indica el nombre de un fichero de texto que contiene la información completa de un objeto. La primera línea del archivo contiene la cadena **"FICHEROVUELO"**. A continuación vendría la información del objeto, con el formato comentado en el ejercicio del operador de salida. Puede asumir que dispone del operador de entrada en la clase **Trayectoria**.

◁ Ejercicio 4 ▷ Métodos de cálculo y sobrecarga de operador > [1.75 puntos]

1. *(0.5 puntos)* Implemente el método llamado **calcularLongitud** que determine la longitud de una trayectoria. La longitud de una trayectoria se calcula sumando la distancia euclídea entre cada par de puntos que la definen. Dados dos puntos (x_1, y_1, z_1) y (x_2, y_2, z_2) la longitud entre ellos se calcula de la siguiente forma:

$$l = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

   Tomando el método anterior como base implemente los métodos **calcularLongitudOptima** y **calcularLongitudReal**.
2. *(0.75 puntos)* Implemente un método denominado **calcularDistanciaMediaTrayectorias** que determine la distancia media de las distancias mínimas de cada punto de la trayectoria real a la trayectoria óptima. La forma de obtener esta medida consiste en determinar, para cada punto p_i de la ruta real el punto más cercano de la ruta óptima, p_c. Sea d_i la distancia entre p_i y p_c. Esta distancia d_i se usará para calcular la distancia media entre trayectorias.
3. *(0.5 puntos)* Sobrecargue el operador **<** en la clase **Vuelo**, usando como criterio la comparación de las distancias medias entre trayectorias de ambos objetos obtenidas mediante el método de cálculo indicado en el punto anterior de esta pregunta.

◁ Ejercicio 5 ▷ Aplicación [1 punto]

Escriba un programa *completo* que reciba como argumento dos nombres de archivo de texto y un valor de tipo **double** como argumento, que contendrán la descripción de dos objetos de la clase **Vuelo**, con el formato considerado en preguntas previas y un valor umbral de distancia. El programa debe tener la siguiente funcionalidad:

1. Calculará el menor vuelo (indique su dato **idVuelo**).
2. Para el vuelo menor se mostrarán longitud real, longitud de ruta óptima y toda la información asociada a sus trayectorias.
3. Indicación de la condición de cercanía entre rutas real y óptima: la distancia media entre las trayectorias es menor o no que el valor umbral de distancia.

Nota: puede convertir una cadena tipo C a un valor double mediante la función **atof**, que recibe como argumento una cadena tipo C (por ejemplo "0.8") y devuelve el valor real asociado. Suponemos que se dispone de los métodos de acceso necesarios para los datos miembro de la clase **Vuelo**.