UNIVERSIDAD DE MÁLAGA

Laboratorio de Programación 2
diciembre 2006
(1º de Ingeniería de Telecomunicación)
E.T.S.I. TELECOMUNICACIÓN

Alumno: Grupo: Máquina:

La clase `CRuta` implementa la gestión de rutas turísticas por una determinada zona. Las rutas constan de una serie de puntos de interés. La zona en la que se localizan las rutas ha sido cuadriculada como una rejilla de tamaño `NF_REJILLA` × `NC_REJILLA` con coordenadas x e y. Cada posición de interés de una ruta viene determinada por su localización, en coordenadas x e y, y su naturaleza: pueden ser restaurantes, gasolineras o áreas de descanso.

Además de los métodos públicos del interfaz de la clase, se muestran los métodos privados que se deben usar como métodos auxiliares en la implementación de la clase.

```
#ifndef __CLASE__CRUTA__
#define __CLASE__CRUTA__

class CRuta
  public:
  // Constructor por defecto: crea una ruta vacía, sin puntos
  CRuta();

  // Destructor
  ~CRuta();

  // Añade puntos a la ruta, que obtiene del fichero cuyo nombre se pasa como parámetro.
  // Cada línea del fichero tendrá el formato: x y tipo, donde 'x' e 'y' representan
  // coordenadas de la rejilla y 'tipo' puede ser 'r' (restaurante), 'g' (gasolinera) ó 'a'
  // (área de descanso). Los puntos se añaden al final de la ruta que ya hubiera.
  // Una  vez añadidos los puntos, deben eliminarse los puntos repetidos.
  void AnyadePuntos(const char nombre[]);

  // Pinta por pantalla la ruta, dibujando para cada elemento de la rejilla un '-'
  // si en esa posición no hay ningún punto de la ruta o, si lo hay, un número
  // que representa la posición que ocupa el punto en la ruta.
  // Debajo se escribirán cuáles son los puntos que corresponden a cada categoría,
  // primero los restaurantes, después las gasolineras y al final las áreas de descanso,
  // indicando el ordinal de los puntos de cada tipo.
  // Por ejemplo, para una rejilla de 6x6 y la ruta
  // {(1,3,r), (2,2,g), (5,1,r), (6,3,a), (4,6,r), (2,5,g)} se pinta en pantalla:
  // --1---
  // -2--6-
  // ------
  // -----5
  // 3-----
  // --4---
  // Restaurantes: 1 3 5
  // Gasolineras: 2 6
  // Areas de descanso: 4
  void PintarRuta() const;

  // Elimina de la ruta los puntos cuya distancia euclídea con el siguiente punto
  // en la ruta sea inferior a 'distancia'. La eliminación debe preservar el orden
  // de los elementos que permanecen en la ruta.
  void PurgaDistanciasCortas(const float distancia);
```

```
  // Enlaza la ruta que se pasa como parámetro con la ruta actual. Para ello, se
  // busca el punto de otraRuta que esté más cercano (en distancia euclídea) con
  // el punto final de la ruta actual. La otraRuta se enlaza con la actual a partir
  // de ese punto más cercano y siguiendo en orden con sus siguientes puntos de
  // manera circular. Se añaden sus puntos hasta el final y se sigue desde el primero
  // hasta llegar otra vez al más cercano.
  void EnlazaRuta(const CRuta &otraRuta);

  private:

  // Constantes privadas. A definir por el alumno.

  // Tipos privados. A definir por el alumno.

  // métodos privados

  // Añade un punto al final de la ruta (puede estar repetido)
  void AnyadeUnPunto(const unsigned int x, const unsigned int y, const TpClasePunto tipo);

  // Elimina de la ruta los puntos repetidos
  void PurgarRepetidos();

  // Devuelve la posición del punto (x,y) ruta, si está, y de qué
  // tipo es: gasolinera, ciudad o área de descanso. Si no, devuelve 0
  unsigned int TipoDePunto(const unsigned int x, const unsigned int y, TpClasePunto &tipo) const;

  // Devuelve la distancia euclídea de dos puntos:
  // ((x1-x2)^2 + (y1-y2)^2)^(1/2)
  float DistanciaEuclidea(const unsigned int x1, const unsigned int y1,

      const unsigned int x2, const unsigned int y2) const;

  // Atributos. A definir por el alumno.
;
#endif
```

Notas

- Se debe trabajar en el directorio /home/alumno/lp2dic06/. Si la ruta no existe, se creará.
- Todos los ficheros de código fuente deben comenzar por un comentario donde se indiquen nombre, apellidos y grupo.
- Esos mismos datos se mostrarán en pantalla al inicio de la ejecución del programa.
- Se deben entregar los siguientes ficheros fuente en un disquete etiquetado con el nombre, apellidos, grupo y número de ordenador: `cruta.hpp`, `cruta.cpp` y `cliente.cpp`.
- Para aprobar es necesario conseguir un programa ejecutable y codificar correctamente el constructor, el destructor y los métodos `AnyadePuntos` y `PintarRuta`. Los métodos `PurgaDistanciasCortas` y `EnlazarRutas` son para subir nota.