Guion de prácticas
El problema del viajante de comercio
Parte 3

Metodología de la Programación
Grado en Ingeniería Informática
Curso: 2016-2017

El Problema del Viajante de Comercio

El objetivo de esta práctica es resolver el problema del viajante de comercio (TSP, por travelling salesmen problem) usando la heurística del vecino más cercano y trabajar la sobrecarga de operadores.

Parte 3

Heurística del vecino más cercano

Esta heurística trabaja de manera iterativa agregando una ciudad al recorrido en cada iteración. Requiere definir dos aspectos: cuál es la ciudad inicial, y qué ciudad se agrega al recorrido. La implementación a realizar procederá de la siguiente manera: dada una ciudad inicial c_0, se agrega como ciudad siguiente aquella c_i (*aún no visitada*) que se encuentre más cercana a c_0. El procedimiento se repite hasta que todas las ciudades se hayan incluido.

Se pide implementar una función externa que reciba como parámetros un objeto de la clase `Problema` y una ciudad y devuelva un objeto de tipo `Solucion` y la longitud del recorrido obtenido. Se proponen las siguientes cabeceras entre las que se deberá decidirse por una o proponer una alternativa:

```
void VecinoMasCercano(Problema p, int inicio, Solucion & sol, int & long)

void VecinoMasCercano(const Problema & p, int inicio, Solucion & sol, int & long)

Solucion VecinoMasCercano(const Problema & p, int inicio, int & long)
```

donde `p` es el problema a resolver, `inicio` es el índice de la ciudad inicial c_0 en la lista de ciudades, `sol` es el recorrido obtenido y `long` es la longitud de dicho recorrido Decida la cabecera más adecuada y justifique su decisión en el código. La función trabajará internamente con un vector de enteros para crear el recorrido y al final, construirá un objeto de tipo `Solucion` que debe devolver de acuerdo a la definición de la función.

Clase Solución

Se pide implementar la sobrecarga del operador de asignación. Depurar y completar el interfaz básico de la clase: añadir los métodos de acceso y asignación necesarios, asegurarse de que las cabeceras son correctas (métodos const, parámetros con los tipos adecuados, const y referencias, ...), usar métodos privados para no repetir código, etc.

Clase Problema

Se pide implementar la sobrecarga del operador de asignación. Depurar y completar el interfaz básico de la clase: añadir los métodos de acceso y asignación necesarios, asegurarse de que las cabeceras son correctas (métodos const, parámetros con los tipos adecuados, const y referencias, ...), usar métodos privados para no repetir código, etc.

Programa Principal

Implemente un programa `tspParte3.cpp` que

1. cree un objeto de la clase `Problema` a partir de un nombre de fichero,
2. llame a la función `VecinoMasCercano` para cada posible ciudad de inicio,
3. debe ir guardando la mejor solución encontrada y su longitud,
4. muestre por pantalla la mejor solución encontrada, mostrando para cada ciudad del recorrido su índice y sus coordenadas y finalmente la longitud del recorrido.

1. Recomendaciones

Antes de empezar a implementar se debe pensar y analizar detenidamente el funcionamiento de la heurística del vecino más cercano.

Utilice los conceptos vistos en teoría y en las sesiones de prácticas: interfaz pública y privada de cada módulo, compilación separada, fichero makefile, etc, organizando todo lo necesario en un directorio que se llamará `tspParte3` con subdirectorios `src, include, obj, bin y doc`.

2. Material a Entregar

Cuando esté todo listo y probado, deberá empaquetar la estructura de directorios en un archivo con el nombre `tspParte3.zip` y lo entregará en la plataforma decsai en el plazo indicado. No deben entregarse archivos objeto (.o) ni ejecutables (conviene ejecutar make clean antes de proceder al empaquetado).

La pareja de estudiantes deben escribir un informe donde consten los nombres y DNI de los integrantes del grupo, los problemas que hayan podido surgir durante el desarrollo de la práctica, capturas de pantalla, etc. Este informe, en formato pdf, se guardará en la carpeta doc. El alumno debe asegurarse de que ejecutando las siguientes órdenes se compila y ejecuta correctamente su proyecto:

```
unzip tspParte3.zip
make
bin/tspParte3 prueba.tsp
```

La ejecución de
`valgrind --leak-check=full --track-origins=yes bin/tspParte3 prueba.tsp`
debe finalizar con 0 errores.