**PRÁCTICA FINAL:**

**RUTAS AÉREAS**

Juan Manuel Rodríguez Gómez

Doble Grado en Ingeniería Informática y Matemáticas

Estructuras de Datos

Curso 2020 – 2021

**1. Implementación de las clases Punto, Ruta y Almacen\_Rutas**

**1.1. Clase Punto**

A continuación, se muestra la parte privada de la clase *Punto*:

Texto

Descripción generada automáticamente

donde, *latitud* y *longitud* representan las coordenadas en el mapa del punto y *descripcion* representa una pequeña explicación asociada a aquellos “puntos de interés”.

Por otro lado, en la parte pública de la clase, se han implementado los siguientes métodos, donde cada método viene acompañado de sus comentarios en Doxygen, explicando principalmente qué hace cada método:

**Texto

Descripción generada automáticamente**

Nótese que al final de la parte pública de la clase se observan dos funciones “amigas” de dicha clase. Dichas funciones son la sobrecarga de los operadores de entrada/salida para un objeto de la clase *Punto*:

Texto

Descripción generada automáticamente

**1.2. Clase Ruta**

La parte privada de la clase *Ruta* es la siguiente:

Texto

Descripción generada automáticamente

Hemos elegido usar el tipo de dato *list* de la STL para así poder insertar puntos tanto al principio como al final de la ruta, consiguiendo así una mayor flexibilidad a la hora de añadir/eliminar puntos de la ruta.

En la parte pública de la clase, se han implementado los siguientes métodos, donde, al igual que ocurre en la clase Punto, cada método viene acompañado de sus comentarios en Doxygen:

Texto

Descripción generada automáticamente

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

Al igual que en la clase *Punto*, al final de la parte pública de la clase se observan dos funciones “amigas”, las cuales son la sobrecarga de los operadores de entrada/salida para un objeto de la clase *Ruta*:

Texto

Descripción generada automáticamente

También se ha implementado, en la parte pública de la clase, dos clases llamadas *iterator* y *const\_iterator*. Estas clases se realizan con el fin de recorrer un objeto de la clase *Ruta*, que recordemos que no es más que una lista de objetos de la clase *Punto*.

Texto

Descripción generada automáticamente

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**1.3. Clase Almacen\_Rutas**

Finalmente, hemos implementado la clase *Almacen\_Rutas*, cuya parte privada es la siguiente:

Texto

Descripción generada automáticamente

Para esta clase, hemos elegido usar el tipo de dato *map* de la STL. De esta forma, cada ruta estará identificada a una sola clave. En este caso, las claves serán datos de tipo *string*.

En la parte pública de la clase, se han implementado los siguientes métodos acompañados de su respectiva documentación en Doxygen:

Texto

Descripción generada automáticamente

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

Como en las clases *Punto* y *Ruta*, se observa al final de la clase la sobrecarga de los operadores de entrada/salida para un objeto de la clase *Almacen\_Rutas*:

**Texto

Descripción generada automáticamente**

Cabe destacar que también hemos tenido que sobrecargar el operador de lectura para un tipo de dato *pair* de la STL que contiene un *string* y un objeto de la clase *Ruta*:

Texto

Descripción generada automáticamente

Como se ha hecho en la clase *Ruta*, hemos implementado, en la parte pública de la clase, dos clases llamadas *iterator* y *const\_iterator* con el fin de recorrer un objeto de la clase *Almacen\_Rutas*:

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**2. Ejemplo de compilación y ejecución del programa**

**2.1. Compilación**

Para compilar el programa disponemos del siguiente *makefile*:

Texto

Descripción generada automáticamente

**Texto

Descripción generada automáticamente**

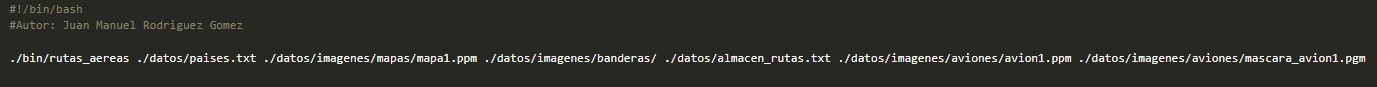
A continuación, se muestra cómo se compila el programa en la terminal de Linux usando el *makefile* anterior:

Texto

Descripción generada automáticamente

**2.2. Ejecución**

Una vez compilado el programa, procedemos a ejecutarlo. Para ello, hemos creado el siguiente *script*:



Finalmente, ejecutamos el programa en la terminal de Linux usando el *script* anterior. Nos mostrará las diferentes rutas y tendremos que elegir una. En este ejemplo hemos elegido la ruta R2.

**Texto

Descripción generada automáticamente**

Obtendremos como resultado la imagen *R2.ppm*, la cual se tiene que encontrar en el directorio */rutas\_aereas* y su contenido es el siguiente:

Mapa

Descripción generada automáticamente

**3. Observaciones y dificultades de la práctica**

Cabe destacar que no se ha modificado el código proporcionado por el profesorado para realizar la práctica, solo se ha comentado con Doxygen.

Hay métodos de las clases implementadas (la clase *Punto*, la clase *Ruta* y la clase *Almacen\_Rutas*) que realmente no se acaban utilizando en esta práctica, pero se han añadido para aportar una mayor funcionalidad a las clases por si se utilizan en la realización de otro programa.

Finalmente, ha habido una dificultad a la hora de realizar la práctica. Dicha dificultad consistía en que, a la hora de ejecutar el script del programa, había veces que sí funcionaba (mostraba las rutas y pedía introducir una) y otras que no (al lanzar el comando *./ejecución.sh* no mostraba nada y se quedaba como en un “bucle infinito”). Tras ejecutar Valgrind para ver cuál podía ser el fallo y dónde se encontraba, obtuvimos lo siguiente:

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Nos dice que hay un salto condicional que depende de un valor no inicializado en la implementación de la sobrecarga del operador de lectura de la clase *Ruta*. La implementación de dicho operador era la siguiente:

Texto

Descripción generada automáticamente

Para solucionar dicho problema cambiamos la línea “*int* num\_puntos” por “*int* num\_puntos = 0”, es decir, había que inicializar la variable de tipo entero *num\_puntos*.

Texto

Descripción generada automáticamente

Si ahora ejecutamos Valgrind, veremos que el problema anterior se ha resuelto y que el programa siempre se ejecuta perfectamente:

Texto

Descripción generada automáticamente