# Práctica Grafos

Estructuras de Datos

## Implementación de la unidad grafo

Crearemos la unidad `UGrafo` que implemente el TAD **Grafo**. El grao que esperamos utilizar en esta ocasión es **disperso**, **conexo** y **no dirigido**. El alumno, con estos datos, debe decidir cuál es la implementación más adecuada. La interfaz de la unidad debe permitir la construcción de cualquier grafo. Se sugiere que la unidad grafo disponga de, al menos, la siguiente funcionalidad accesible en su interfaz:

```
PROCEDURE CrearGrafoVacio(VAR g:TGrafo);
PROCEDURE InsertarOrigen(VAR g:TGrafo; origen:TElemento);
PROCEDURE InsertarDestino(g:TGrafo; origen:TElemento; destino:
TElemento);
FUNCTION EsGrafoVacio(g:TGrafo):Boolean;
FUNCTION PerteneceAOrigenes(g:TGrafo; origen:TElemento):Boolean;
FUNCTION PerteneceADestinos(g:TGrafo; origen:TElemento; destino:
TElemento):Boolean;
PROCEDURE ListaAdyacencia(g:TGrafo; e:TElemento; VAR adyacentes:TLista);
```

## Uso de la unidad grafo

El Metro de Madrid puede representarse como un grafo, en el que las estaciones son los vértices y las conexiones entre ellas, las aristas. Por ejemplo, una estación como Sol, tiene seis estaciones adyacentes: Ópera, Sevilla, Callao, Lavapies, Gran Vía y Tirso de Molina (las dos primeras son de la línea 2, las dos siguiente de la línea 3 y las dos últimas de la línea 1). Se puede consultar un plano del metro en la siguiente URL: https://www.metromadrid.es/export/sites/metro/comun/documentos/planos/Planoesquematicoespanol.pdf

Para probar la funcionalidad de nuestra implementación del TAD grafo, vamos a modelar con ella el Metro de Madrid. Para ello, se proporcionan 12 ficheros de texto (desde L01.txt hasta L12.txt, en formato UTF8, de modo que si utilizas windows, quizá necesites paarlos a formato ASCII), uno por cada línea de metro, en los que se listan las estaciones que las componen, una por cada línea del fichero. Se debe observar que las líneas circulares (6 y 12) aparecen con estructura lineal en el archivo, pero nos resultarán igualmente útiles de esa manera.

Se pide implementar la siguiente funcionalidad:

1.- *Construir el grafo del Metro de Madrid*: para ello, se debe leer la lista de estaciones de cada archivo proporcionado. Cada una de las estaciones debe tener al menos los datos de su nombre y el/los número/s identificativo/s de la/s línea/s a la/s que pertenece. Para la construcción del grafo, recuerda que debes utilizar **exclusivamente** la interfaz de la unidad `Ugrafo`.

2.- *Consultar estaciones*: desde el programa principal, permitir al usuario realizar consultas por estaciones. Para cada estación, se debe mostrar si existe o no en el grafo y, en caso de existir, las líneas a las que pertenece y las estaciones adyacentes.

3.- *Recorrido*: Implementar en la unidad `UGrafo` un recorrido en anchura desde un origen dado. El recorrido en anchura de un grafo parte de un vértice dado `A` (cualquiera) y explora en primer lugar todos los vértices adyacentes. Luego toma uno de esos vértices y explora todos sus adyacentes no visitados previamente, luego desde otro adyacente del origen `A` ya visitado en primera iteración se exploran todos sus adyacentes no visitados en pasos anteriores y así sucesivamente. Para mantener memoria de los vértices que debemos

visitar en cada paso se utilizará una **estructura FIFO**, y para evitar entrar en ciclos y revisitar vértices ya visitados previamente se utilizará un **conjunto** de vértices visitados. Seguro que ya tienes implementadas estas estructuras auxiliares. Si es así, utiliza esas implementaciones, recordando ser muy cuidadoso y acceder a su funcionalidad exclusivamente desde la interfaz proporcionada.

4.- *Camino*: Implementar también en la unidad `UGrafo` la funcionalidad necesaria para el cálculo de un camino entre un vértice origen y un vértice destino (no importa si es de longitud mínima o no).

Nota: Como siempre, se hará uso de las normas de estilo dictadas en clase (cabecera del fichero, interfaz de la unidad con precondiciones, postcondiciones, excepciones, implementaciones con el análisis de complejidad de cada operación, nombres coherentes de variables y operaciones,...)

Plantilla de cabecera del fichero:

```
{*****************************************************************************
*       Módulo:                                                                 *
*       Tipo:   Programa()     Interfaz-Implementación TAD ()        Otros()    *
*       Autor/es:                                                               *
*       Fecha de actualización:                                                 *
*       Descripción:                                                            *
*****************************************************************************}
```