Punto en el plano - Trayectorias

Se define una **trayectoria** como una **secuencia de puntos no repetidos**.

Representaremos una trayectoria mediante un **array de estructuras Punto**.

Primera ampliación

Ampliar el programa **punto2D** para que …

- Permita trabajar con trayectorias de hasta 25 puntos (definir este valor como constante en el programa).
- Genere una trayectoria de puntos, asignando las coordenadas del punto de forma aleatoria. El valor de cada coordenada deberá estar comprendido entre -1000,00 y 1000,00 (definir estos valores, tipo real, como constantes en el programa).

  En la generación deberá garantizarse que no existirán puntos repetidos en la trayectoria.

  Los valores de las coordenadas generadas para los puntos deberán ser reales.
- Permita visualizar en pantalla una trayectoria.

Requisitos que deben de cumplirse en el desarrollo
Se desarrollarán las siguientes funciones:

- void **CrearTrayectoria** (Punto *tr, int dim, int linf, int lsup);
  - ***Punto *tr*** y ***int dim*** representan la trayectoria, es decir, el array de estructuras Punto.
  - ***float linf*** y ***float lsup*** representan el valor mínimo y máximo de las coordenadas de cada punto, a generar aleatoriamente.
  - En la generación deberá garantizarse que no existirán puntos repetidos en la trayectoria.

- void **VerTrayectoria** (const Punto *tr, int dim);
  - ***Punto *tr*** y ***int dim*** representan la trayectoria.
  - Deberá presentarse en pantalla cada Punto de la trayectoria.

Segunda ampliación

Para cualificar una trayectoria, calcularemos tres parámetros asociados a la trayectoria:

- La distancia mínima.
- La distancia máxima.
- La distancia media.

Los anteriores valores se calculan considerando todas las distancias existentes, comparando cada punto con el resto, dentro de la trayectoria.

Por ejemplo, si la trayectoria constara sólo de tres puntos, p1, p2 y p3, entonces:

- Distancia mínima será la distancia más pequeña de entre d(p1, p2), d(p1, p3) y d(p2, p3).
- Distancia máxima será la distancia más grande de entre d(p1, p2), d(p1, p3) y d(p2, p3).
- Distancia media será (d(p1, p2)+d(p1, p3)+d(p2, p3)) / 3

Para almacenar la cualificación de una trayectoria, definiremos un nuevo tipo estructura como el siguiente:

```
typedef struct  {
   float dmedia;
   float dminima;
   float dmaxima;
} CualificadorTr;
```

Realizar un programa que

- Permita cualificar una trayectoria.
- Muestre en pantalla los tres parámetros que cualifican a la trayectoria.

Requisitos que deben de cumplirse en el desarrollo
Se desarrollarán las siguientes funciones:

- float **DistanciaMediaTrayectoria** (const Punto *tr, int dim);
- float **DistanciaMinimaTrayectoria** (const Punto *tr, int dim);
- float **DistanciaMaximaTrayectoria** (const Punto *tr, int dim);
  - Las tres anteriores funciones realizarán el cálculo que su nombre indica.
  - Sólo deberán ser llamadas desde la función **CualificarTrayectoria** que se describe a continuación.

- void **CualificarTrayectoria** (const Punto *tr, int dim, CualificadorTr *ct);
  - Llamando a las funciones **DistanciaMediaTrayectoria, DistanciaMinimaTrayectoria** y **DistanciaMaximaTrayectoria**, cargará los tres miembros de la estructura **CualificadorTr** que recibe como parámetros por referencia.

- void **VerCualificadorTrayectoria** (const CualificadorTr *ct);
  - Esta función presentará en pantalla las tras cantidades que cualifican una trayectoria.
  - El cualificador de la trayectoria se recibe como argumento en la función.

Ayuda

Pseudocódigo para …

Almacenar m elementos en un array vx, garantizando que los elementos "no estarán repetidos"

```
vx[1] ← elemento calculado/leido

Desde i ← 2 hasta i=m con intremento 1

  Repetir
      repetido ← FALSO // flag
      vx[i] ← elemento calculado/leido
      k ← i-1
      Mientras ((k >= 1) AND NOT repetido) {
          Si (vx[i] = vx[k])
              repetido ← CIERTO
          FinSi
          k ← k-1
      FinMientras

  Mientras (repetido)

FinDesde
```