PROGRAMACIÓN ORIENTADA A OBJETOS

GRADO EN INGENIERÍA INFORMÁTICA
SEGUNDO CURSO

DEPARTAMENTO DE INFORMÁTICA Y ANÁLISIS NUMÉRICO
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CÓRDOBA

CURSO ACADÉMICO: 2012 – 2013

- **Número de práctica: 2**

- **Objetivos**
  - Definir clases que permitan manejar tipos abstractos de datos utilizando el punto de vista de "la programación orientada a objetos".
    - Clase Punto2D
    - Clase Recta2D

    - Conceptos básicos de las clases en C++
      - Parte pública y parte privada
      - Atributos
      - Funciones o métodos
      - Constructores
        - Constructor parametrizado con valores por defecto
        - Constructor de copia
      - Calificador const en los métodos de acceso
      - Funciones auxiliares de una clase.
    - Utilización de los comentarios de doxygen

- **Descripción**
  - Se deben codificar las funciones de las clases Punto2D y Recta2D para que funcione correctamente el programa codificado en el fichero "practica_2.cpp"
  - Las clases y las funciones auxiliares se deben integrar en el espacio de nombres "poo" para el código desarrollado en la asignatura.
  - Se deben codificar los siguientes ficheros
    - punto2D.hpp
    - punto2D.cpp
    - recta2D.hpp
    - recta2D.cpp
    - makefile

- **Fichero punto2D.hpp**

- **Declaración de la clase Punto2D**

  - **Atributos privados**
    - Coordenadas "x" e "y" del punto en el plano euclídeo P = (x, y)
    - Las coordenadas son del tipo real de punto flotante.

  - **Funciones o métodos públicos**
    - Constructor: **Punto2D**
      - Versión 1: **constructor parametrizado con valores por defecto**
        - Recibe como parámetros a las coordenadas "x" e "y" y crea un objeto del tipo Punto2D con las coordenadas indicadas.
        - Esta función tiene argumentos por defecto.
        - El valor por defecto de los parámetros es 0.0
      - Versión 2: **constructor de copia**
        - Recibe como parámetro un Punto2D "q" pasado por referencia **constante** y crea otro Punto2D con una copia de los valores del parámetro "q".

    - Consulta
      - **getX**: devuelve el atributo "x" de un Punto2D
      - **getY**: devuelve el atributo "y" de un Punto2D
      - **Aviso**: estas funciones deben utilizar el calificador **const**

    - Modificación
      - **setX**:
        - Recibe un valor numérico "v" y se lo asigna a la componente "x" del Punto2D.
      - **setY**:
        - Recibe un valor numérico "v" y se lo asigna a la componente "y" del Punto2D.

    - Funciones de lectura o escritura
      - **leerPunto2D**
        - Lee desde el teclado las coordenadas y se las asigna al Punto2D.
      - **escribirPunto2D**:
        - Escribe por pantalla las coordenadas del Punto2D de la forma "(x, y)".

    - **Observación**:
      - Las únicas funciones que podrán acceder directamente a los datos privados de Punto2D serán las funciones de acceso (getX y getY) y de modificación (setX y setY).
      - Las demás funciones podrán acceder a los datos privados a través de las funciones de acceso y modificación
      - Las versiones del constructor y las funciones de consulta y modificación se codificarán **"inline"**.

- **Función auxiliar de la clase Punto2D**
  - **Observación**
    - Esta función no pertenece a la clase
  - **calcularDistanciaEuclidea2D**
    - Recibe dos Puntos2D y devuelve su distancia euclídea.
    - Si P1 = (x1, y1) y P2 = (x2, y2) entonces
      d(P1,P2) = $\sqrt{(x2-x1)^2+(y2-y1)^2}$

- **Fichero punto2D.cpp**
  - Debe contener el código de las siguiente funciones
    - Funciones de lectura y escritura de la clase Punto2D
      - leerPunto2D
      - escribirPunto2D
    - Función auxiliar de la clase Punto2D
      - calcularDistanciaEuclidea2D
    - Se recuerda que la función auxiliar también pertenece al espacio de nombre "poo"

- **Fichero recta2D.hpp**

  - **Declaración de la clase Recta2D**
    - **Atributos privados**
      - Coeficientes "a", "b" y "c" de la recta en el plano euclídeo.
        - a X + b Y + c = 0
      - Los coeficientes son del tipo real de punto flotante.

    - **Funciones o métodos públicos**
      - Constructor: **Recta2D**
        - Versión 1: **constructor parametrizado con valores por defecto**
          - Recibe como parámetros a los coeficientes "a", "b" y "c" y y crea un objeto del tipo Recta2D con los coeficientes indicados.
          - Esta función tienen argumentos por defecto.
          - El valor por defecto de los coeficientes es 1.0, que permitirá crear la recta X + Y + 1 = 0
        - Versión 2: **constructor de copia**
          - Recibe como parámetro una Recta2D "s" pasada por referencia **constante** y crea otra Recta2D con una copia de los valores del parámetro "s".
        - Versión 3:
          - Recibe dos Puntos2D "p1" y "p2" y crea la recta que pasa por dichos puntos.

- Consulta
  - **getA**: devuelve el atributo “a” de la Recta2D.
  - **getB**: devuelve el atributo “b” de la Recta2D.
  - **getC**: devuelve el atributo “c” de la Recta2D.
  - **Aviso**: estas funciones deben utilizar el calificador **const**

- Modificación
  - **setA**: recibe un valor numérico “v” y le asigna dicho valor al coeficiente “a” de la Recta2D.
  - **setB**: recibe un valor numérico “v” y le asigna dicho valor al coeficiente “b” de la Recta2D.
  - **setC**: recibe un valor numérico “v” y le asigna dicho valor al coeficiente “c” de la Recta2D.

- Funciones de lectura o escritura
  - **leerRecta2D**
    - Lee desde el teclado los coeficientes y se los asigna a la Recta2D.
  - **escribirRecta2D**:
    - Escribe por pantalla la ecuación de la Recta2D
      “a X + b Y + C = 0”.
    - Se valorarán las siguientes mejoras
      - Escribir correctamente el signo de los coeficientes
      - Omitir los coeficientes nulos.
      - Si un coeficiente vale 1 entonces solamente se ha de escribir la incógnita
        - Por ejemplo
          r: 3 x - 1 y + 0 = 0
        - Se podría escribir
          r: 3 x - y = 0
- **Observación:**
  - Las únicas funciones que podrán acceder directamente a los datos privados de Recta2D serán las funciones de acceso (getA, getB y getC) y de modificación (setA, setB y setC).
  - Las demás funciones podrán acceder a los datos privados a través de las funciones acceso y modificación.
  - Las versiones del constructor y las funciones de consulta y modificación se codificarán “**inline**”.

- **Funciones auxiliares de la clase Recta2D**
  - **Observación**
    - Estas funciones no pertenecen a la clase
  - **calcularDistanciaPunto2DRecta2D**
    - Recibe un Punto2D “p” y una Recta2D “r” y calcula la distancia del punto a la recta
      - Observación
        - Si el punto p = (x0, y0) y la recta es $aX + b\ Y + c = 0$ entonces la distancia del punto a la recta es

$$d(p,r) = \frac{|a\,x0 + b\,y0 + c|}{\sqrt{a*a + b*b}}$$

- **sonRectas2DPerpendiculares**
  - Recibe dos Rectas2D “r1” y “r2” comprueba si son perpendiculares
  - Las rectas
    - r*1: a1 X + b1 Y + c1 = 0*
    - *r2: a2 X + b2 Y + c2 = 0*
      son perpendiculares si *a1 * a2 + b1 * b2 = 0.0*
  - El valor devuelto ha de ser de tipo “bool”

- **sonRectas2DParalelas**
  - Recibe dos Rectas2D “r1” y “r2” comprueba si son paralelas
  - Las rectas
    - *r1: a1 X + b1 Y + c1 = 0*
    - *r2: a2 X + b2 Y + c2 = 0*
      son paralelas si *a1 * b2 - b1 * a2 = 0.0*
  - El valor devuelto ha de ser de tipo “bool”

- **Fichero recta2D.cpp**
  - Debe contener el código de las siguiente funciones
    - Funciones de lectura y escritura de la clase Recta2D
      - leerRecta2D
      - escribirRecta2D
    - Funciones auxiliares de la clase Recta2D
      - calcularDistanciaPunto2DRecta2D
      - sonRectas2DPerpendiculares
      - sonRectas2DParalelas
    - Se recuerda que las funciones auxiliares también pertenecen al espacio de nombre “poo”