Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

ESTRUCTURAS DE DATOS

GRADO EN INGENIERÍA INFORMÁTICA
Segundo curso. Segundo cuatrimestre.

Curso académico 2016 – 2017

PRIMERA PRÁCTICA

POLÍGONO DE VÉRTICES DEL PLANO EUCLIDIANO

- **Objetivo**
  - Implementar el tipo abstracto de datos **Poligono2D** utilizando **dos** representaciones internas diferentes:
    - mediante un **vector** de STL
    - mediante una **lista** de STL

- **Descripción de las clases**
  - Clase **Vertice2DInterfaz**
    - Clase abstracta que especifica los métodos “virtuales puros” para acceder y modificar los atributos de un vértice del plano euclidiano: abscisa y ordenada.
    - Nota: esta clase ya está codificada en el fichero vertice2DInterfaz.hpp
  - Clase **Vertice2D**
    - Clase que representa un vértice de un polígono en el plano euclidiano.
    - Hereda de forma pública de la clase **Vertice2DInterfaz**
    - Nota: esta clase ya está codificada en los ficheros vertice2D.hpp y vertice2D.cpp
    - Atributos:
      - Abscisa y ordenada del vértice
      - Son datos de tipo real
    - Métodos:
      - Constructores
        - Constructor con todos los argumentos con valores por defecto
        - Constructor de copia
      - Observadores o funciones de consulta de los atributos
      - Modificadores de los atributos
      - Operadores
        - Operador de asignación “=”: hace una copia de un vértice
        - Operador de igualdad “==”: compara dos vértices (tiene en cuenta la precisión de los números reales).
      - Funciones de lectura y escritura
    - Además, también se proporcionan las siguientes funciones auxiliares que no pertenecen a la clase Vertice2D
      - calcularDistanciaEuclidiana2D
      - funciones **amigas** para sobrecargar los operadores de flujo “>>” y “<<”

- Clase **Poligono2DInterfaz**
  - Clase abstracta
  - Se debe codificar el fichero poligono2DInterfaz.hpp para especificar los siguientes métodos “virtuales puros” de un polígono del plano euclidiano.
    - Consultar el número de vértices del polígono
      - Función de tipo “const”
    - Comprobar si un polígono es nulo o vacío
      - Función de tipo “const”
    - Comprobar si existe el vértice que ocupa la posición señalada por un índice
      - Función de tipo “const”
    - Acceder al vértice que ocupa la posición señalada por un índice
      - Función de tipo “const”
    - Insertar un vértice al final del polígono
    - Borrar un vértice de un polígono situado en una posición señalada por un índice
- Clase **Poligono2D**
  - Clase que hereda de forma pública de la clase Poligono2DInterfaz
  - Se deben codificar los ficheros poligono2D.hpp y poligono2D.cpp para especificar y definir los atributos y los métodos de un polígono del plano euclidiano.
  - Atributo: vértices
    - Primera implementación: **vector STL** de objetos de la clase **Vertice2D**
    - Segunda implementación: **lista STL d**e objetos de la clase **Vertice2D**
  - Nota: cada implementación se deberá codificar en un directorio diferente
    - Version-Vector
    - Version-Lista
  - Métodos
    - Constructor
      - Constructor sin argumentos: crea un polígono vacío con cero vértices.
      - Constructor de copia: crea un polígono a partir de otro
    - Destructor
    - Observadores o funciones de consulta
      - Comprobar si el polígono está vacío
      - Número de vértices del polígono
      - Comprobar si existe un vértice
      - Acceder al vértice que ocupa la posición “i”:
        - Se deberán codificar dos versiones
          - Uso de la función: getVertice2D
          - Uso del operador []
    - Modificadores
      - Insertar un vértice al final del polígono
      - Borrar el vértice situado en la posición señalada por un índice
      - Borrar el polígono, es decir, todos su vértices
    - Operadores
      - Operador de asignación “=”: hace una copia de un polígono
      - Operador de igualdad “==”: compara dos polígonos
    - Entrada y salida

- Leer un polígono desde el teclado
- Escribir un polígono por pantalla
- Cargar un polígono desde un fichero de texto
- Grabar un polígono en un fichero de texto

- Funciones auxiliares
  - Calcular el centroide del polígono
    - El centroide es el centro de gravedad y sus coordenadas se obtienen mediante la media aritmética de las coordenadas de los vértices del polígono:
      - $x = \frac{1}{N} \sum_{i=1}^{N} x_i$
      - $y = \frac{1}{N} \sum_{i=1}^{N} y_i$
      - donde N es el número de vértices del polígono
      - y las coordenadas del vértice i-ésimo son (x_i, y_i)
  - Calcular el perímetro
    - El perímetro de un polígono es la suma de sus lados, es decir, la suma de las distancias euclidianas entre sus vértices.
  - Calcular el área
    - Se debe utilizar la "fórmula determinante de Gauss"

      $$A = \left| \frac{1}{2} \sum_{i=0}^{N-1} (x_i (y_{i+1} - y_{i-1})) \right|$$

    - donde
      - N es el número de vértices del polígono
      - (x_i, y_i) i =0, 1, 2,… , N -1
      - Los vértices están numerados de forma cíclica, es decir,
        - $(x_{-1}, y_{-1}) = (x_{N-1}, y_{N-1})$
        - $(x_N, y_N) = (x_0, y_0)$
  - Ejemplo: abrelatas.txt
    - Centroide: (357.04, 271.383)
    - Perímetro: 662.843
    - Área: 13219

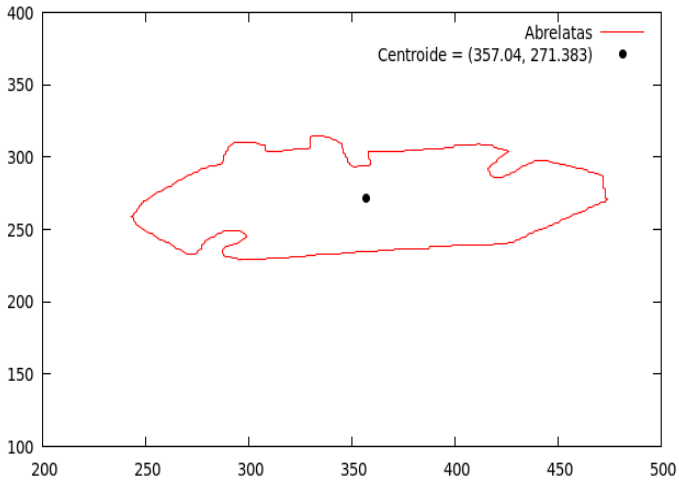

Figura. Polígono del abrelatas generado con gnuplot

- **Descripción de los ficheros**
  - **Introducción**
    - Se proporciona un fichero comprimido denominado “practica-1.zip” con dos directorios
      - Version-Vector
      - Version-Lista
    - Cada directorio contiene los ficheros que se describen a continuación.
    - El estudiante debe completar el código de los ficheros que se indican en cada caso.
  - **makefile**
    - Facilita la compilación de los ficheros, la generación de la documentación con doxygen y el borrado de ficheros que ya no sean necesarios (*.o, *~, etc.)
    - El estudiante puede mejorar este fichero.
  - **Doxyfile**
    - Fichero de configuración para generar la documentación con doxygen
    - El estudiante puede modificar este fichero para mejorar su documentación.
  - **principalPolinomio2D.cpp**
    - Programa principal utilizado como ejemplo para comprobar el funcionamiento de la clase Poligono2D
    - Fichero proporcionado por el profesor
    - Utiliza macros de pantalla para mejorar la visualización de la información.
    - El estudiante puede ampliar y mejorar este programa de ejemplo.
  - **macros.hpp**
    - Fichero con macros para mejorar la visualización de la información en la pantalla.
  - **funcionesAuxiliares.hpp**
    - Incluye los prototipos de funciones auxiliares utilizadas en el programa principal del fichero principalPolinomio2D.cpp
  - **funcionesAuxiliares.cpp**
    - Código complementario de las funciones auxiliares utilizadas en el programa principal
    - Importante: el estudiante debe completar este fichero.
  - **poligono2DInterfaz.hpp**
    - Definición de la clase abstracta Poligono2DInterfaz
    - Importante: el estudiante debe completar este fichero.
  - **poligono2D.hpp**
    - Definición de la clase Poligono2D que hereda de la clase Poligono2DInterfaz
    - Importante: el estudiante debe completar este fichero.
  - **poligono2D.cpp**
    - Código complementario de las funciones de la clase Poligono2D
    - Importante: el estudiante debe completar este fichero.
  - **vertice2DInterfaz.hpp**
    - Definición de la clase abstracta Vertice2DInterfaz
  - **vertice2D.hpp**
    - Definición de la clase Vertice2D que hereda de la clase Vertice2DInterfaz
  - **vertice2D.cpp**
    - Código complementario de las funciones de la clase Vertice2D
  - **Ficheros con polígonos de ejemplo**
    - triangulo.txt, cuadrado.txt y abrelatas.txt

- **Observaciones sobre la entrega de la práctica número 1**
  - Duración de la práctica 1: tres sesiones de dos horas cada una.
  - Plazo máximo de entrega
    - Grupos de los lunes: 27 de febrero de 2017
    - Grupos de los martes: 7 de marzo de 2017
    - Grupo del jueves: 2 de marzo de 2017
  - Se deberá subir un fichero comprimido denominado “practica-1-usuario.zip”, donde “usuario” es el login de cada estudiante.
  - Este fichero comprimido contendrá dos directorios
    - Version-Vector: implementación utilizando un vector de la STL
    - Version-Lista: implementación utilizando una lista de la STL
  - Nota:
    - Se debe usar el espacio de nombres de la asignatura: **ed**

- **Evaluación**
  - La calificación de la práctica se basará
    - en la calidad del trabajo realizado
    - y en su defensa **presencial**.
  - Se valorará
    - El correcto funcionamiento del programa principal propuesto como ejemplo:
      - Véase el fichero principalPolinomio2D.cpp
    - La ampliación y mejora del menú del programa principal para añadir más opciones.
    - La documentación del código con doxygen
    - La claridad del código
    - El uso de macros de pantalla para mejorar la visualización de la información