Tarea 1: Introducción a C++

EP7007-1 – Computación en GPU

Profesora: Nancy Hitschfeld-Kahler
Auxiliar: Sergio Salinas
Ayudante: Andrés Abarca

Abril 2023

1 Introducción

Con objetivo de aprender/practicar programación orientada a objetos en c++, en esta tarea, se pide implementar tres clases para representar objetos geométricos simples. Las clases son `Punto`, `Vector` y `Poligono`, diseñar algunos tests para probar que funcionan y crear un main pequeño que cree instancias de estas clases y le aplique los métodos programados. Cada clase debe tener una serie de métodos y atributos que se detallan en las siguientes secciones.

Clase Punto

Implementar una clase `Punto` que represente un punto en un espacio bidimensional utilizando templates. La clase debe tener los siguientes miembros:

- Una variable privada tipo `std::pair` para guardar dos coordenadas x e y.
- Un constructor que reciba dos valores de tipo `T` y los asigne a la variable privada correspondiente.
- Métodos públicos para calcular la distancia entre este punto y otro.
- Operadores sobrecargados para las siguientes operaciones:
  - Comparación: `bool operator==(const Punto<T>& other)`.
  - Salida de datos en el formato `(x,y)` al operador `std::ostream& operator<<(std::ostream& os, const Punto<T>& p)`.

Clase Vector

Implementar una clase `Vector`, representa un vector matemático (no confundir con el vector de C++) en un plano cartesiano en dos dimensiones, utilizando templates. Esta clase debe tener los siguientes miembros:

- Dos variables privadas para almacenar las coordenadas `x` e `y` del punto usado para representar el vector.
- Constructor que inicializa un vector con sus coordenadas 'x' e 'y' de tipo `T`.
- Constructor que inicializa un vector en base a un punto
- Método que calcule la magnitud de un vector.
- Método que calcule el producto punto entre dos vectores
- Método que calcule el producto cruz entre dos vectores
- Operadores sobrecargados para las siguientes operaciones:
  - Suma `Vector<T> operator+(const Vector<T>& other)`.
  - Comparación: `bool operator==(const Vector<T>& other)`.
  - Producto por escalar: `Vector<T> operator*(T scalar)`.
  - Salida de datos en el formato `(x,y)` al operador `std::ostream& operator<<(std::ostream& os, const Vector<T>& p)`.

Clase Poligono

Implementar una clase `Poligono` que represente un polígono en un espacio bidimensional utilizando templates. La clase debe tener los siguientes miembros:

- Un vector (estructura de datos) privado que almacene los puntos del polígono.
- Un entero privado que indique la cantidad de puntos del polígono.
- Un constructor que reciba un vector (estructura de datos) de puntos y los agregue al polígono.
- Método para obtener la cantidad de puntos del polígono.
- Método que indique si un polígono esta en counterclockwise o no
- Operadores sobrecargados para las siguientes operaciones:
  - Un operador sobrecargado que retorne el punto del polígono en la posición indicada: `Punto<T>& operator[](int i)`.
  - Salida de datos en el formato `[(x1,y1),(x2,y2),...,(xn,yn)]` para el operador: `std::ostream& operator<<(std::ostream& os, const Poligono<T>& p)`.

2 Observaciones

- Las clases punto y vector deben llevar templates.
- Se pueden agregar los constructores que sean necesarios, si es necesario, también destructor.
- Se pueden agregar métodos y atributos extras de ser necesario.
- Se deben agregar tests, para ellos se puede la librería CASSERT. No es necesario que se usen librerías adicionales.
- No se tomaran en cuenta métodos sin tests. **De no haber test en toda la tarea, se calificara con la nota mínima.**
- Se puede usar un makefile para compilar la tarea, no es necesario el uso del CMake
- Incluir readme que indique como ejecutar la tarea.