1. Se desea representar un polígono con una interface simple para poder conocer algunas propiedades del mismo. La representación e interface deseada es la siguiente. Se solicita completar la implementación del UDT.

```
struct Punto2D {
    double X;
    double Y;
};

struct Poligono {
    public:
        // construye un poligono cerrado con los vertices dados
        Poligono(const vector<Punto2D> &vertices);

        // retorna el perimetro del poligono
        double perimetro();

        // retorna el area del poligono
        double area();

    private:
        vector<Punto2D> vertices;
};
```

   Evalúe la necesidad de implementar un constructor de copia, uno de asignación y un destructor. En el caso de llegar a la conclusión de que hacen falta, impleméntelos.

2. **Islas de grafeno**: Para depositar grafeno sobre una superficie metálica se utiliza la técnica CVD (Chemical Vapor Deposition) donde se inserta un sustrato metálico y se modula el flujo de gases reactivos en una cámara a alta temperatura. Experimentalmente se ha detectado que durante la deposición de grafeno sobre superficies metálicas se forman islas cuyo número y tamaño depende de las presiones parciales de los gases. Por medio de STM (Microscopio Electrónico de Barrido por efecto túnel) se pueden obtener imágenes en escala de grises con intensidades entre 0 y 1023. Se desea implementar un software que, a partir de una imagen, determine automáticamente las islas de grafeno presentes.
   El controlador del STM almacena imágenes en un canal de 10 bits sin comprimir. Esto es, una imagen en escala de grises con intensidades entre 0 y 1023.

```
struct STM_image {
    unsigned int width;
    unsigned int height;
    vector<unsigned short> pixels;

    unsigned int ij2index(int i, int j) {
        assert(i >= 0 && i < (int) width && j >= 0 && j < (int) height);
        unsigned int pixelIndex = i * width + j;
        assert(pixelIndex < pixels.size());
        return pixelIndex;
    }

    unsigned short getPixel(int i, int j) {
        unsigned int pixelIndex = ij2index(i, j);
        return pixels[pixelIndex];
    }

    void setPixel(int i, int j, unsigned short value) {
        unsigned int pixelIndex = ij2index(i, j);
        pixels[pixelIndex] = value;
    }
};
```

Dado que la densidad de estados del grafeno es sensiblemente diferente a la del metal, el controlador puede realizar un pre-procesamiento de la imagen y resaltar las islas de grafeno con mayor intensidad que el metal.
Una manera simplificada de caracterizar una isla es a través de los puntos que la forman:

```
struct Punto2D {
    int x, y;
};

struct IslaGrafeno {
    vector<Punto2D> puntos;
};
```

Se desea diseñar e implementar un **`GrafenoAnalizer`**: un tipo que posee un método **`analize`** que recibe una **`STM_image`** como argumento y termina retornando el vector de islas (**`vector<IslaGrafeno>`**) presentes en la imagen recibida.

```
struct GrafenoAnalizer {
    GrafenoAnalizer(unsigned short threshold_) {
    threshold = threshold_;
}

vector<IslaGrafeno> analize(STM_image img);
// ...
    unsigned short threshold;
};
```

Para encontrar las islas se puede recorrer la imagen pixel a pixel. Si el valor de un pixel [Fila, Columna] es mayor que un nivel de disparo dado threshold, entonces se ha encontrado una isla: se procede a encontrar todos sus puntos almacenándolos en una IslaGrafeno y se continúa recorriendo el vector de pixeles.

HINT: Una vez ubicada una isla piense en utilizar un algoritmo parecido al FloodFill visto en la práctica para ir encontrando los puntos, pintándolos como para evitar que vuelva a ser encontrados y agregándolos al vector que definirá la isla.