examenparcialpracticasL5def.pdf

_juanitoo03

Fundamentos de Programación

1° Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería
Universidad de Huelva

Grado en Ingeniería Informática. **Fundamentos de Programación.**
Evaluación continua. **Segunda prueba práctica**.
Grupo L5.
21 de Enero de 2021.

- La prueba tiene una duración de 2 horas (incluido el tiempo para acceder al examen y la subida del resultado a *moodle*).
- Se aplicará una penalización del 30% sobre la nota obtenida por el alumno en caso de que el programa proporcionado como solución al problema no compile correctamente.
- Es necesario que el alumno incluya su nombre y correo electrónico en un comentario dentro del propio código del archivo *.cpp* que se entregará como solución al problema.

Ejercicio. Se va a realizar un programa en C++ para realizar operaciones con triángulos definidos a través de las coordenadas de sus vértices en un espacio bidimensional. Con ese objetivo, y dada la declaración del registro Punto propuesta a continuación, implemente la clase Triangulo de acuerdo con las siguientes características:

```
struct Punto { // representa un punto en un espacio bidimensional, a través de sus coordenadas en los ejes x e y.
        int coordX;
        int coordY;
};
```

Atributos privados:

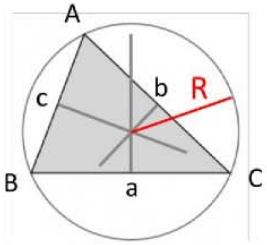
- Punto A, B, C; // representan las coordenadas de los tres vértices que definen el triángulo.
- char etiqueta[30]; // cadena de caracteres que almacena el nombre con que se etiqueta el triángulo en el sistema.

Métodos públicos:

- Un constructor sin parámetros
  /* Inicializa los atributos *A, B* y *C* con las coordenadas de los puntos (0, 0), (1, 0) y (0, 1) respectivamente. Inicializa *etiqueta* con la cadena de caracteres vacía*/
- Un constructo parametrizado Triangulo (int aX, int aY, int bX, int bY, int cX, int cY, char nombre[30]);
  /* Inicializa *A* asignando respectivamente *aX* y *aY* a sus campos *coordX* y *coordY*. Inicializa *B* asignando respectivamente *bX* y *bY* a sus campos *coordX* y *coordY*. Inicializa *C* asignando respectivamente *cX* y *cY* a sus campos *coordX* y *coordY*. Inicializa *etiqueta* con el valor de la cadena *nombre*. */
- Un constructo parametrizado Triangulo::Triangulo(punto aP, punto bP, punto cP, char nombre[30]);
  /* Inicializa los atributos *A, B y C* con los parámetros *aP*, *bP* y *cP* respectivamente. Inicializa *etiqueta* con el valor de la cadena *nombre*. */
- bool Triangulo::setVertice(char c, punto p);
  /* Modifica el vértice indicado por el parámetro *c*, asignándole *p*. Si *c* es el carácter 'A' el atributo modificado será *A*. Si *c* es el carácter 'B' el atributo modificado será *B*. Si *c* es el carácter 'C' el atributo modificado será C. Si *c* es cualquier otro carácter ningún atributo será actualizado al valor de *p*. El método devolverá *true* si se ha realizado la modificación de algún atributo; *false* en caso contrario. */
- Punto getVerticeA();
  /* Devuelve el atributo *A*/

- Punto getVerticeB();
  /* Devuelve el atributo *B*/
- Punto getVerticeC();
  /* Devuelve el atributo *C*/
- void setEtiqueta(char nombre[30]);
  /* Modifica el atributo *etiqueta*, asignándole el valor del parámetro *nombre*. */
- void getEtiqueta(char nombre[30]);
  /* Devuelve el atributo *etiqueta* a través del parámetro de entrada/salida *nombre*. */
- double calculaCircunradio ();
  /* Calcula la longitud del circunradio del triángulo y la devuelve como resultado. */

El circunradio de un triángulo es el radio la circunferencia circunscrita a dicho triángulo. Dicha circunferencia es aquella cuyo centro (circuncentro) es el punto en donde intersectan las mediatrices asociadas a cada uno de los lados del triángulo, y que equidista de sus tres vértices. El circunradio *R* de un triángulo puede calcularse conocida la longitud de todos sus lados (*a*, *b* y *c*), a partir de la siguiente expresión:



$$R = \frac{a \times b \times c}{4 \times \sqrt[2]{s \times (s-a) \times (s-b) \times (s-c)}}$$

$$s = {}^{(a+b+c)}/_{2}$$

Siendo *a*, *b* y *c* la longitud de los tres lados del triángulo, y *s* el semiperímetro del mismo.

Por otra parte, la longitud de un lado del triángulo es igual a la distancia euclídea entre los vértices que lo delimitan. La distancia euclídea entre dos puntos, *P1* y *P2*, definidos en un espacio bidimensional a partir de las coordenadas cartesianas (x_1, y_1) y (x_2, y_2) se obtiene a partir de la siguiente expresión:

$$d(P1, P2) = \sqrt[2]{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Implemente a continuación las siguientes funciones:

bool buscaTriangulo(Triangulo vTriangulos[], int n, int &posición, char nombre[30]);

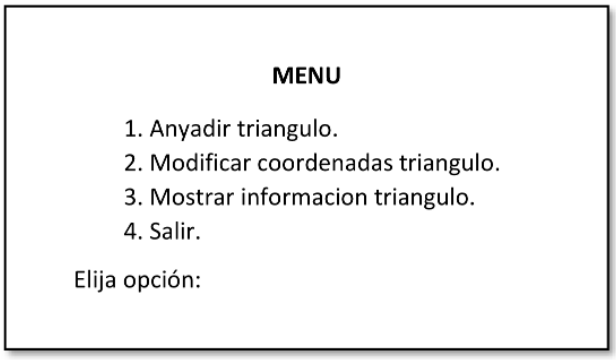
/* Recibe un vector de objetos de la clase *Triangulo* (*vTriangulos*), el número de objetos contenidos en dicho vector (*n*) y una cadena de caracteres (*nombre*). Devolverá *true* si encuentra en el vector *vTriangulos* un objeto de la clase Triangulo cuya etiqueta coincide con el parámetro *nombre*, indicando en el parámetro de entrada/salida *posicion* la posición de dicho objeto en el vector. Si no encuentra ningún objeto, devolverá *false*, y un -1 en el parámetro de entrada/salida *posicion* */

void imprimirInformacionTriangulo(Triangulo t);

/* Mostrará por pantalla de forma ordenada la etiqueta de *t*, las coordenadas de cada uno de sus vértices y la longitud de su circunradio. El formato de representación es libre, pero toda esta información ha de ser fácilmente identificable, y mostrada de forma clara y concisa. */

int menuTriangulos ();

/* Muestra el siguiente menú, y devuelve la opción seleccionada. */



Finalmente implemente una función principal *main()* que compruebe el correcto funcionamiento de la clase. Para ello se definirán las siguientes variables locales:

Triangulo triangulos[MAX_TRIANGULOS];
/* vector de objetos de la clase *Triangulo,* donde MAX_TRIANGULOS será una constante a definir, con valor 100.*/
int nTriangulos = 0; // número de elementos del vector *triangulos*.

La función *main()* mostrará el menú propuesto a través de la función *menuTriangulos()* y realizará las siguientes acciones según la opción elegida:

1) Si hay espacio en el vector *triangulos* solicitará por teclado un nombre (etiqueta) para el nuevo triángulo a introducir en el vector. Si ya existe un triángulo con dicha etiqueta, mostrará un mensaje de error. En caso contrario solicitará por teclado las coordenadas de los tres vértices del triángulo, y se actualizará el vector *triangulos* incluyendo un nuevo objeto con la información introducida por el usuario. Si no hay espacio suficiente en el vector, se mostrará un mensaje de error.

2) Se solicitará al usuario que introduzca la etiqueta del triángulo a modificar. Si no se encuentra ningún objeto dentro del vector *triangulos* con dicha etiqueta, se mostrará un mensaje de error. En caso contrario, una vez encontrado el triángulo a actualizar, se solicitará por teclado el vértice a modificar. Éste será indicado con un carácter: 'A' para modificar el vértice *A*, 'B' para el *B*, y 'C' para el *C*. Seguidamente se solicitará las nuevas coordenadas del vértice. Se intentará la modificación del vértice a través del método de la clase apropiado, notificando al usuario si la actualización se ha realizado con éxito, o bien no ha sido posible al no especificarse de forma adecuada el vértice a modificar.

3) Se solicitará al usuario que introduzca la etiqueta del triángulo a buscar. Si éste no se encuentra en el vector *triangulos*, se mostrará un mensaje de error. En caso contrario se mostrará por pantalla toda la información característica de dicho triángulo, esto es, su etiqueta y las coordenadas de cada uno de sus vértices, así como la longitud de su circunradio.

4) Saldrá del programa.

Nota:

- Está permitido el uso de las funciones definidas en las librerías *string,h* y *math.h*.
- Está permitida la implementación y uso de funciones auxiliares.