classe Position2D

L'objectif de ce premier TP consistera à déclarer et à utiliser une classe, `Position2D`, représentant une position dans le plan avec les coordonnées x et y entières et positives ou nulles.

Condition impérative : cette classe ayant pour but de représenter des positions sur l'écran, en aucun cas, les coordonnées ne doivent pouvoir être négatives !

Le but final, à la fin du TP, sera d'écrire une classe qui implémentera les fonctionnalités suivantes:

1. un constructeur par défaut qui initialise les coordonnées à (0, 0)
2. un constructeur permettant de spécifier les deux coordonnées cartésiennes en assurant la condition ci-dessus, prototype: **`Position2D::Position2D(int, int)`**
3. un constructeur permettant de spécifier deux valeurs représentant les coordonnées polaires de la position, *r* et θ en assurant la condition ci-dessus, prototype: **`Position2D::Position2D(double, double)`**
4. une méthode qui écrit sur la console la position sous la forme (x, y); ex: **`(12, 56),`** prototype: **`void Position2D::affiche(void)`**
5. une méthode qui renvoie la distance euclidienne d'une position à une autre position spécifiée en argument, prototype: **`double Position2D::distance(Position2D);`**
6. une méthode qui renvoie le milieu d'une position et d'une autre position spécifiée en argument, prototype: **`Position2D Position2D::milieu(Position2D);`**
7. la surcharge de l'opérateur + qui renvoie une position de coordonnées x et y respectivement égales à la somme des coordonnées de la position avec une autre spécifiée en argument? prototype: **`Position2D Position2D::operator +(Position2D);`**
8. une méthode rotation qui applique à la position une rotation autour de l'origine d'un angle exprimé en degrés, donné en argument - la condition doit être respectée ici aussi; prototype: **`void Position2D::rotation(int);`**

Marche à suivre

En utilisant l'environnement **geany**, créer un nouveau fichier ("nouveau selon un modèle", puis "main.cxx") que vous sauverez sous le nom de tp1-votre_nom.cpp[1] à un endroit du disque où vous saurez le retrouver.

Inclure la déclaration de la classe au dessus de la fonction principale (main), comme expliqué en cours et l'implémentation des méthodes de la classe en fin de fichier (après main).

travail à réaliser	
déclarer la classe Position2D et implémenter uniquement les méthodes des points 1 et 4 ci-dessus (constructeur par défaut et affiche). Écrire un programme principal (main.cpp) qui instancie quatre objets, p1, p2, p3 et p4, de la classe Position2D puis les affiche. Vérifier le fonctionnement dans le programme principal.	
rajouter et implémenter les constructeurs des points 2 et 3, puis les utiliser pour deux des trois objets précédents. Vérifier le fonctionnement dans le programme principal.	
rajouter et implémenter les méthodes des points 5 à 8, puis les utiliser dans le programme principal en vérifiant à chaque fois votre bonne compréhension des choses, et le fonctionnement correcte des méthodes.	
Après avoir dupliqué votre programme (enregistrer sous…) dans un autre fichier, remplacer	

1 attention, il faut impérativement que l'extension soit .cpp, et non .c !

votre programme principal par le suivant :

```
int main(int argc, char **argv)
{
Position2D p1, p2(67, -20), p3(100.0, 0.707), p4;                 // quatre positions

// affichage des quatre positions initiales
cout<<"p1:";p1.affiche();
cout<<"p2:";p2.affiche();
cout<<"p3:";p3.affiche();
cout<<"p4:";p4.affiche();

double d = p2.distance(p3);
cout<<"distance p2 p3:"<<d<<endl;
p4 = p1.milieu(p2);
p1 = p2+p4;
p2 = p1.rotation(10);

cout<<"après traitement"<<endl;
cout<<"p1:";p1.affiche();
cout<<"p2:";p2.affiche();
cout<<"p3:";p3.affiche();
cout<<"p4:";p4.affiche();

return 0;
}
```

Si vous avez bien respecté le cahier des charges, vous devez impérativement obtenir :

```
p1:(0,0)
p2:(67,0)
p3:(76,64)
p4:(0,0)
distance p2 p3:64.6297
après traitement
p1:(100,0)
p2:(98,17)
p3:(76,64)
p4:(33,0)
```

Chaque binôme devra envoyer par email une archive (.zip) contenant le ou les fichiers constituant le programme complet du TP, et ce, impérativement avant le jour de la séance suivante.

Si le programme qui ne se compile pas (présence d'erreurs de syntaxe) la note de compte rendue ne sera donnée que sur la moitié des points prévus.

Si vous n'avez pas pu terminer le programme durant la séance, charge à vous de le terminer durant la semaine.

La présentation et les commentaires dans les programmes seront très appréciés.