Marseille, 04/10/2019 - Achref El Mouelhi

TP 1 : POO avec C++ (partie 1)

Exercice 1

Considérons une classe C++ appelée `Point` ayant les attributs suivants :

- `abs` : un attribut privé de type double
- `ord` : un attribut privé de type double

1. Créez les deux fichiers `Point.h` et `Point.cpp`
2. Définissez des getters et setters, `inline`, pour les deux attributs.
3. Définissez deux constructeurs, `inline`, sans et avec deux paramètres : `Point()` et `Point(double abs, double ord)`.
4. Définissez une méthode `afficher` qui permet d'afficher les coordonnées d'un point sous le format suivant : `(abs, ord)`.
5. Écrivez la méthode `calculerDistance(Point p)` qui permet de calculer la distance entre le point de l'objet courant (`this`) et l'objet `Point p` passé en paramètre. Nous rappelons que la distance entre deux points `p1`(x_1,y_1) et `p2`(x_2,y_2), en mathématiques, est égale à : $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2)}$
   Utiliser la fonction `sqrt(double a)` définies dans `cmath`
6. Définissez une méthode statique `distance(double` x_1`, double` y_1`, double` x_2`, double` y_2`)` qui calcule et retourne la distance entre les deux points A(x_1,y_1) et B(x_2,y_2).
7. Écrivez la `calculerMilieu(Point p)` qui permet de calculer et de retourner un objet correspondant au milieu du segment défini par le point de l'objet courant (`this`) et l'objet `Point p` passé en paramètre. Nous rappelons que les coordonnées d'un point `m`(x_M,y_M) milieu de `p1`(x_1,y_1) et `p2`(x_2,y_2), en mathématiques, sont :
   - $x_M = \frac{x_1 + x_2}{2}$
   - $y_M = \frac{y_1 + y_2}{2}$

   La méthode doit retourner un objet `Point` et pas les coordonnées.
8. Refaire la question précédente en surchargeant les deux opérateurs `+` et `/` qui permettrons d'avoir le point milieu de la manière suivante : `Point m = (p1 + p2) / 2`

Considérons maintenant une deuxième classe appelée `TroisPoints` ayant les attributs suivants :

- `premier` : un attribut privé de type `Point`
- `deuxième` : un attribut privé de type `Point`
- `troisième` : un attribut privé de type `Point`

9. Créez les deux fichiers `TroisPoints.h` et `TroisPoints.cpp`.
10. Définissez les getters/setters et le constructeur avec trois paramètres de la classe `TroisPoints` en `inline`.

11. Écrivez une méthode `testerAlignement()` qui retourne `true` si les trois points `premier`, `deuxième` et `troisième` sont alignés, `false` sinon. Nous rappelons que trois points A, B et C sont alignés si AB = AC + BC, AC = AB + BC ou BC = AC + AB (AB désigne la distance séparant le point A du point B, pareillement pour AC et BC).
12. Écrivez une méthode `estIsocele()` qui retourne `true` si les trois points `premier`, `deuxième` et `troisième` forment un triangle isocèle, `false` sinon. Nous rappelons qu'un triangle ABC est isocèle si AB = AC, AB = BC ou BC = AC.
13. Dans la fonction principale `int main()`, demandez à l'utilisateur de saisir les coordonnées de trois points. Ensuite, utilisez les classes et les méthodes précédentes pour afficher tous les détails sur ces trois points, les milieux, les distances qui les séparent, afficher s'ils sont alignés, s'ils forment un triangle isocèle...

Exercice 2

Considérons une classe C++ appelée `Stagiaire` ayant les attributs suivants :

- `nom` : un attribut privé de type chaîne de caractère
- `notes` : un attribut privé de type tableau

La classe `Stagiaire` reçoit la taille du tableau grâce à un template de classe.

1. Créez la classe `Stagiaire`
2. Définissez des getters et setters pour les attributs.
3. Définissez deux constructeurs sans et avec deux paramètres `Stagiaire(string nom, int notes [])`
4. Écrivez la méthode `calculerMoyenne()` qui permet de retourner la moyenne des notes d'un stagiaire
5. Écrivez les méthodes `trouverMax()` et `trouverMin()` qui permettent de retourner respectivement les notes max et min d'un stagiaire.
6. Redéfinissez l'opérateur >= comme fonction amie dans `Stagiaire.h` et implémenter la dans `main.cpp` pour permettre la comparaison de la moyenne de deux stagiaires. La valeur de retour de cet opérateur est un booléen.

Considérons maintenant une classe appelée `Formation` (avec template) ayant les attributs suivants :

- `intitulé` : un attribut privé de type chaîne de caractère
- `nbrJours` : un attribut privé de type entier
- `stagiaires` : un tableau d'objets de type `Stagiaire` chacun avec 3 notes

7. Créez la classe `Formation`, préparez les getters et setters de ses attributs, et définissez deux constructeurs avec et sans paramètres `Formation(string intitulé, int nbrJours, Stagiaire stagiaires [])`
8. Écrivez une méthode `calculerMoyenneFormation()` qui retourne la moyenne d'un objet de type formation (la moyenne des moyennes des stagiaires)
9. Écrivez une méthode `getIndexMax()` qui retourne l'indice du stagiaire dans le tableau `stagiaires` ayant la meilleure moyenne de la formation.
10. Écrivez une méthode `afficherNomMax()` qui affiche le nom du premier stagiaire ayant la meilleure moyenne d'une formation.

11. Écrivez une méthode `afficherMinMax()` qui affiche la note minimale du premier stagiaire ayant la meilleure moyenne d'une formation.
12. Écrivez une méthode `trouverMoyenneParNom(string nom)` qui affiche la moyenne du premier stagiaire dont le nom est passé en paramètre.
13. Dans la fonction principale `main`, testez toutes les méthodes réalisées dans les questions précédentes (créez par example trois objets `Stagiaire`, affectez les à une même formation et faites appel à toutes les méthodes que vous avez implémentées).

Exercice 3

Considérons les deux classes C++ `Personne` et `Adresse`. Les attributs de la classe `Adresse` sont :

- `rue` : un attribut privé de type chaîne de caractère.
- `ville` : un attribut privé de type chaîne de caractère.
- `codePostal` : un attribut privé de type chaîne de caractère.

Les attributs de la classe `Personne` sont :

- `nom` : un attribut privé de type chaîne de caractère.
- `sexe` : un attribut privé de type caractère (cet attribut aura comme valeur soit `'M'` soit `'F'`).
- `adresses` : un attribut privé de type tableau d'objet de la classe `Adresse`.

1. Créez les deux classes `Adresse` et `Personne` dans deux fichiers séparés. N'oubliez pas de définir les getters/setters et les constructeurs.
2. Créez une troisième classe `ListePersonnes` ayant un seul attribut `personnes` : un tableau de maximum 10 objets de la classe `Personne`. Créer les getters/setters et le constructeur de cette classe.
3. Écrivez la méthode `findByNom(string s)` qui permet de chercher dans le tableau `personnes` si un objet dont le nom égal au paramètre `s` existe. Si c'est le cas, elle retourne le premier objet correspondant, sinon `null`.
4. Écrivez la méthode `findByCodePostal(string cp)` qui permet de vérifier dans le tableau `personnes` si un objet possède au moins une adresse dont le code postal égal au paramètre `cp`. Si c'est le cas, elle retourne `true`, sinon `false`.
5. Écrivez la méthode `countPersonneVille(string ville)` qui permet de calculer le nombre d'objets dans le tableau `personnes` ayant une adresse dans la ville passée en paramètre.
6. Écrivez la méthode `editPersonneNom(string oldNom, string newNom)` qui remplace les noms de personnes ayant un nom égal à la valeur `oldNom` par `newNom`
7. Écrivez la méthode `editPersonneVille(string nom, string newVille)` qui remplace les villes de personnes ayant un nom égal à la valeur du paramètre `nom` par `newVille`

Exercice 4

Considérons une classe C++ appelée `MaDate` ayant les trois attributs suivants :

- `jour` : un attribut privé de type entier.
- `mois` : un attribut privé de type entier.
- `année` : un attribut privé de type entier.

1. Créez la classe `MaDate`
2. Définissez une méthode `afficher()` afin que nous puissions afficher une date sous le format `jour/mois/année`.
3. Générez (ou écrivez) les getters et setters des trois attributs.
4. Définissez un constructeur avec trois paramètres `MaDate(int jour, int mois, int année)`
5. Écrivez la méthode `ajouterUnJour` qui permet d'ajouter un jour à notre date et faire des modifications, si nécessaire, pour les deux attributs `mois` et `année`. Attention, il faut traiter tous les cas. Par example si les trois attributs `jour`, `mois` et `année` contiennent respectivement les valeurs 31, 12 et 2016, alors la méthode `ajouterUnJour` doit affecter la valeur 1 à `jour`, 1 à `mois` et 2017 à `année`. Et s'ils contiennent 28, 02 et 2018 alors les nouvelles valeurs après modification seront respectivement 29, 02 et 2018.
6. Utilisez la méthode de la question précédente pour écrire la méthode `ajouterPlusieursJours(int n)` : `n` étant le nombre de jours à ajouter à la date enregistrée dans les trois attributs.
7. D'une façon similaire, définissez les méthodes `ajouterUnMois` et `ajouterUnAn()`.
8. Dans la fonction `main`, testez toutes les méthodes réalisées dans les questions précédentes.