POO sous C++ - TP1

Exercice 1
Donner un programme qui demande le nom, le prénom et l'âge de l'utilisateur ensuite les enregistrer respectivement dans des variables de type **class string** et les affichées comme suit :

-Donner votre nom :
Nom
-Donner votre prénom :
Prénom
- Ton age :
22
Bonjour *Nom Prénom* ton age est *22* ans

Exercice 2
Donner un programme de conversion de longueur **cm** (c) en unité de **pouce** (p) ou l'inverse selon le choix de l'utilisateur. Sachant que le nombre qu'un **pouce** (**inch**) est égale à **2.54 cm** `( 1 i = 2.54 cm)`.
Donner deux proposition une utilisant une instruction **if** et l'autre utilisant l'instruction **switch**.

Exercice 3
La classe **string** vous permet d'accéder aux caractères en utilisant soit l'opérateur [] soit la méthode **at()**. Pour connaître la longueur de la chaîne, vous pouvez utiliser soit **length() soit size()**. En utilisant la classe **string**, donner un programme qui affiche les codes ASCII des caractères composant une chaîne de caractères.

Exercice 4
Ecrire un programme qui demande à l'utilisateur de taper 10 entiers et qui affiche le plus petit de ces entiers.

Exercice 5
Ecrire un programme qui demande à l'utilisateur de taper un entier N et qui calcule u(N) défini par :

```
u(0)=1
u(1)=1
u(n+1)=u(n)+u(n-1)
```

Exercice 6
Ecrire un programme qui permet de faire des opérations sur un entier (valeur initiale à 0). Le programme affiche la valeur de l'entier puis affiche le menu suivant :

1. . Ajouter 1
2. . Multiplier par 2
3. . Soustraire 4
4. . Quitter

Exercice 7
Ecrire un programme qui demande à l'utilisateur de taper des entiers strictement positifs et qui affiche leur moyenne. Lorsqu'on tape une valeur négative, le programme affiche ERREUR et demande de retaper une valeur. Lorsqu'on tape 0, cela signifie que le dernier entier a été tapé. On affiche alors la moyenne. Si le nombre d'entiers tapés est égal à 0, on affiche PAS DE MOYENNE.

Exercice 8
Soit le modèle de structure suivant :

```
struct essai{
int n ;
float x ;
};
```

Écrire une fonction nommée raz permettant de remettre à zéro les 2 champs d'une structure de ce type transmise en argument :

1. par adresse ;
2. par référence.

Dans les deux cas, on écrira un petit programme d'essai de la fonction ; il affichera les valeurs d'une structure de ce type, après appel de ladite fonction.

Exercice 9
Écrire plus simplement en C++ les instructions suivantes, en utilisant les opérateurs new et delete :

```
int * adi ;
double * add ;
.....
adi = malloc (sizeof (int) ) ;
add = malloc (sizeof (double) * 100 ) ;
```

Exercice 10
Réaliser une classe *point* permettant de manipuler un point d'un plan. On prévoira :

- un constructeur recevant en arguments les coordonnées (*float*) d'un point ;
- une fonction membre **deplace** effectuant une translation définie par ses deux arguments (*float*) ;
- une fonction membre **affiche** se contentant d'afficher les coordonnées cartésiennes du point.

Les coordonnées du point seront des membres donnée privés. On écrira séparément :

- un fichier source constituant la **déclaration** de la classe ;
- un fichier source correspondant à sa **définition**.

Écrire, par ailleurs, un petit programme d'essai (*main*) déclarant un point, l'affichant, le déplaçant et l'affichant à nouveau.

Exercice 11
Réaliser une classe point, analogue à la précédente, mais ne comportant pas de fonction affiche. Pour respecter le principe d'encapsulation des données, prévoir deux fonctions membre publiques (nommées abscisse et ordonnee) fournissant en retour l'abscisse et l'ordonnée d'un point. Adapter le petit programme d'essai précédent pour qu'il fonctionne avec cette nouvelle classe.

Exercice 12
Ajouter à la classe précédente (comportant un constructeur et trois fonctions membre *deplace, abscisse* et *ordonnee*) de nouvelles fonctions membre :

- *homothetie* qui effectue une homothétie dont le rapport est fourni en argument;
- *rotation* qui effectue une rotation dont l'angle est fourni en argument;
- *rho* et *theta* qui fournissent en retour les **coordonnées polaires** du point.

Exercice 13
Soit la classe point créée dans l'exercice 10, dont la déclaration était la suivante :

```
class point
{
float x, y ;
public :
point (float, float) ;
void deplace (float, float) ;
void affiche () ;
}
```

Adapter cette classe, de manière que la fonction membre *affiche* fournisse, en plus des coordonnées du point, le nombre d'objets de type *point*.