Université Hassan II de Casablanca
ENSET de Mohammedia
Département Mathématiques et Informatique
Filière d'Ingénieurs : GLSID 1 & BDCC 1
Prof. M. K. MANSOURI

TP 2
PROGRAMMATION ORIENTEE OBJET EN C++
CLASSES, OBJET, CONSTRUCTEURS ET DESTRUCTEURS

Exercice 1 :

Ecrire une classe nommée ***point***, constituées de deux membres données entier x et y et 4 fonction fonctions :

- ***Initialiser*** : fonction à trois paramètres dont la couleur, permettant de prendre des valeurs initiale pour représenter le point d'origine (sous forme d'une étoile),
- ***Deplacer*** : fonction à deux paramètres, permettant de déplacer le point à une nouvelle position,
- ***Afficher*** : fonction sans paramètres, permettant d'afficher le point dans la position courante,
- ***Effacer*** : fonction sans paramètres, permettant d'effacer le point.

Prévoir dans la fonction **main**, des appels aux fonctions membres pour initialiser le point l'afficher, le déplacer et l'effacer plusieurs fois.

Exercice 2 :

Utiliser la classe ***point*** précédente. Ecrire une fonction **Scene** sans paramètres dans laquelle on déclare un point u, on l'initialise, on l'affiche , on le déplace et on l'affiche à nouveau. Le programme principal *main* ne contient que l'appel à *test*.

Exercice 3 :

Reprendre la classe ***point*** précédente en replaçant fonction membre **initialise** par un constructeur.

Exercice 4 :

Modifier l'exercice 3 ci-dessus en utilisant un constructeur à deux paramètres.

Exercice 5 :

Reprendre la classe ***point*** précédente en ajoutant un destructeur qui met fin à la vie de l'objet « dernier point » en affichant ses coordonnées.

Exercice 6 :

Ecrire une classe nommée ***SuiteAr***, dans laquelle le constructeur calcule les premiers (nb) termes d'une suite arithmétique de raison (Nul) qu'il range dans un tableau membre donné val. Le destructeur libère l'espace mémoire réservé pour le membre (Val).

Exercice 7 :

a- Ecrire une classe nommée ***hasard***, dans laquelle le constructeur fabrique dix valeurs entières aléatoires qu'il range dans un tableau membre donné val. Ces valeurs sont prises entre zéro et la valeur qui lui est fourni en argument)

b- On désire disposer d'une classe nommée ***hasard***, dans laquelle le nombre de valeurs peut être fourni en argument du constructeur.
Prévoir une allocation dynamique de val pour qu'il s'adapte automatiquement au nombre de valeurs voulu.
Naturellement cette allocation dynamique doit être faite par le constructeur lui même.

c- Ecrire un destructeur permettant de libérer l'espace alloué.

Exercice 8 :

a- Ecrire une classe ***Complexe*** qui possède deux champs, correspondant aux parties réelle et imaginaire du nombre complexe que l'on souhaite représenter et trois méthodes, la première définit un nombre complexe à partir de ses parties réelles et imaginaires, la seconde calculant le module et la dernière affichant ce nombre complexe à l'écran.

b- Proposer trois constructeurs différents de cette classe que vous écrivez séparément.

c- Proposer trois constructeurs différents de cette classe que vous écrivez séparément.

```
class Complexe
{
    private:
    double x, y;

    public:
    void set(double, double);
    double module();
    double affiche();
    double real() const;
    double img() const;
};
```

```
void Complex::set(double a, double b)
{
    x = a;
    y= b;
}
double Complex::module()
{
   double res;
   res = sqrt(x*x + y*y);
   return res;
}
void affiche()
{
   printf(``%f + %f I\n'', x, y);
}
```

```
Complex::Complex(double a, double b)
{
   x = a; y= b;
}
Complex::Complex(double a)
{
   x = a; y= 0;
}
Complex::Complex()
{
   x = 0; y= 0;
}
```