

Serie N°1

(Nouveautés apportées par C++ par rapport au C)

- (Entrées/sorties en C++)

Ecrire un programme qui permet de saisir les éléments d'un tableau et d'afficher le maximum et le minimum de ce tableau. (En utilisant les flux d'entrées /sortie **cout** et **cin**)

- (surcharge des fonctions)

On souhaite donner le même nom à trois fonctions. La première additionne deux entiers (type int), la deuxième deux réels (type float) et la troisième deux tableaux de dix entiers. Donner le corps de ces fonctions. Puis les appeler dans la fonction principale main()

- (Arguments par défauts)

Considérez une fonction destinée à afficher un entier un nombre de fois que l'utilisateur va choisir :

```
void print(int value, int nbre=10) ; /*le nombre de fois d'affichage par défaut est 10*/  
void f(){  
print(31) ; print(31,10) ; print(31,16) ; print(31,2) ;  
}
```

Écrivez une fonction qui prend 2 entiers en argument (2 entiers auxquels on donnera une valeur par défaut). Et donnez les différents appels possibles de cette fonction.

Faire de même avec une fonction prenant 4 arguments dont 2 seulement sont donnés par défaut.

- (les références)

Ex:1

- o On déclare un tableau d'entiers A de dimension 10
- o Deux fonctions, remplitA et impA, permettent de remplir et imprimer le tableau.
- o Déclarer une référence vers l'élément 5 du tableau et incrémenter sa valeur puis l'afficher
- o Déclarer un pointeur vers l'élément 7, incrémenter le pointeur et la valeur pointée .puis les afficher

Ex:2

- o Déclarer un entier
- o Déclarer une référence vers cet entier
- o Déclarer un pointeur vers cet entier
- o Dans les deux cas, afficher la variable, l'adresse de la variable, la valeur pointée.

Ex:3

Ecrire une fonction qui modifie le jour, le mois et l'année d'une date donnée. Il y a trois solutions à ce problème (passage par valeur, pointeur, et références). Les mettre en œuvre et expliquer les différences.

- (Allocation dynamique de la mémoire)

Créer un tableau de 20 entiers d'une manière dynamique.

Saisir et afficher les valeurs du tableau. puis libérer la mémoire allouée pour ce tableau.

Serie N°2

Initiation aux classes objets en c++

Exercice1:

Créer une classe nommée Point possédant:

- 3 attributs privés:(int x, int y, char *label).
- 3 constructeurs(par défaut, normal et par copie).
- 3 fonctions membres(void afficher(), void déplacer(int,int) et double distance(Point)).

Dans le même projet C++, vous devez avoir les fichiers suivants:

- un fichier nommé **Point.h** pour la déclaration de la classe
- un fichier nommé **Point.cpp** pour la définition de chaque fonction membre
- un fichier nommé **main.cpp** pour faire appel aux fonctions de la classe Point

Exercice2:

Traiter de la même manière que l'exercice précédent la classe suivante:

```
Class Cercle{
int x,y;
int rayon;
char *label;
public:
Cercle();// constructeur par défaut
Cercle(int,int,int,char*);// constructeur normal
Cercle(Cercle &);// constructeur par copie
Void afficher(); // affiche les attribut de la classe
Void déplacer(int dx,int dy);// déplace l'origine du cercle
Void agrandir (int r);// Agrandir le rayon du cercle en lui ajoutant r
Void réduire(int r);// réduire le rayon du cercle en lui diminuant r
double perimetre();// retourne le périmètre du cercle
double surface();// retourne la surface du cercle.
bool comparer(Cercle C); //qui compare la surface de l'objet courant avec celle du cercle
C
};
```

Serie N°3

(Destructeurs, objets membres, membres static et membres constants)

Exercice1:

Dans le même projet:

- Définir une classe Point qui contient 3 attributs: x, y, char *label
- La classe Point ne contiendra pas de constructeurs par défaut
- Ajouter un attribut à la classe point qui permettra de savoir le nombre de point créés.
- Ajouter une fonction la classe point qui permettra de retourner le nombre de point créés.
- Ajouter un destructeur à la classe Point.
- Définir une classe Segment qui contient 3 attributs:origine et extremite qui sont deux objets de type Point, et epaisseur de type entier.
- Ajouter un attribut constant à la classe Segment : int numserie
- Ajouter un constructeur à la classe segment.
- Ajouter une fonction afficher() à la classe Segment.
- Ajouter une fonction longueur() à la classe Segment qui retourne la longueur du segment.
- Ajouter une fonction void milieu(Point &M) qui retrouve le point du milieu du segment et l'enregistre dans M.
- Dans la fonctions principale main():
 - créer deux objets de type point. Afficher leurs attributs
 - créer un segment à partir de ces deux points. Afficher ses attributs.
 - Afficher sa longueur
 - Afficher les attributs de son milieu.
 - créer un troisième point.
 - Afficher le nombre de points créés

Exercice2:

Refaire l'exercice précédent, mais en ajoutant cette fois-ci:

- Un constructeur par défaut à la classe Point.
- Un constructeur normal à la classe Segment: Segment(Point,point,int,int)

Exercice3 :

Créer une classe nommée duree contenant:

- 3 attributs privés(int h,min,s;)
- Constructeurs (tester la validité de la duree s<60 et min<60)
- 4 fonctions: void afficher(); duree additionner(duree);duree multiplier(int); et bool comparer(duree);

Serie N°4

(Fonctions amies, classes amies, surcharge des opérateurs)

Exercice1 (fonction amie) :

- 1- Définir une classe Tableau contenant deux attributs un tableau (int *t) et la taille du tableau (int taille).
- 2- Définir un 3 types de constructeurs
- 3- Définir une fonction membre void Trier() qui trie un tableau dans l'ordre croissant
- 4- Définir une fonction membre void Renverser() qui renverse les éléments du tableau t
- 5- Définir une fonction amie à la classe Tableau void afficher(Tableau &) qui affiche les éléments du tableau
- 6- Définir un destructeur de la classe Tableau

Exercice2 (classe amie) :

- 1- Définir une classe Examen amie de la classe Duree (classe definie dans la serie 3) contenant 3 attributs privés:
 - Un tableau de 30 éléments de type: struct etudiant{char nom[20],prenom[20]; float note}
 - Un attribut d de type Duree qui contient la duree de l'examen.
 - Un attribut char matiere[20] qui contient le nom de la matière.
- 2- Définir un constructeur normal Examen(etudiant *, Duree)
- 3- Une fonction void afficher() qui affiche les attributs de la classe.
- 4- Une fonction int duree_seconde() qui retourne la duree de l'examen en secondes.

Exercice3 (Surcharge des opérateurs) :

Surcharger les opérateurs suivants dans la classe Duree(classe definie dans la serie 3):

- **Operator+** **Duree operator+(const Duree &);**
- **Operator*** **Duree operator*(int nombre);**
- **Operator==** **bool operator==(const Duree &);**
- **Operator<** **bool operator<(const Duree &);**
- **Operator=** **Duree operator=(const Duree &);// affectation**

Exercice4 (Surcharge des opérateurs) :

- 1) Créer une classe Rationnel dont les objets sont des nombres rationnels, avec un numérateur et un dénominateur.
- 2) On déclarera des méthodes pour donner une valeur à un Rationnel, le convertir en nombre réel, l'inverser et l'afficher.
Surcharger les opérateurs suivants dans la classe Rationnel:+,*,==,<