Filière d’ingénieur :

Ingénierie Informatique, Big Data et Cloud Computing (II-BDCC)

Département Mathématiques et Informatique

Compte rendu TP3

DE LA CONCEPTION ET LA PROGRAMMATION

ORIENTE OBJET EN C++

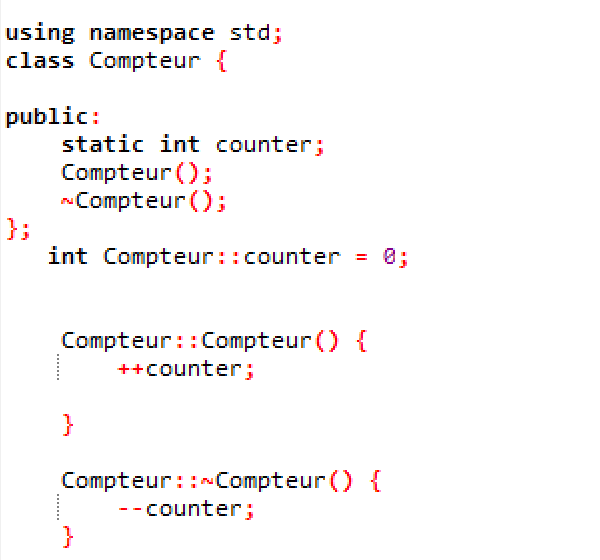
Par: BOUZINE Ahmed houssam (II-BDCC1)

**Encadré par : M. K. MANSOURI**

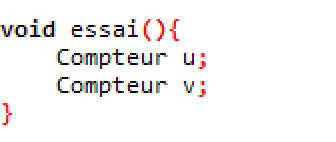
**Année Universitaire : 2022/2023**

Exercice 1 :

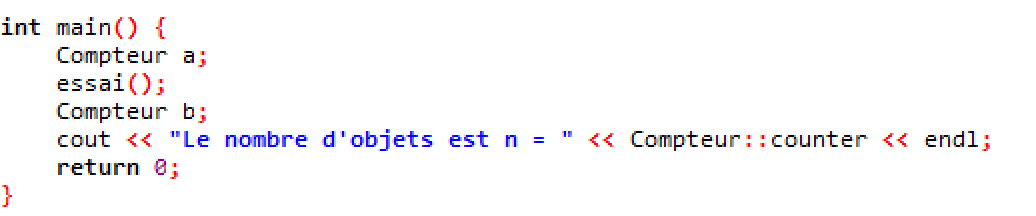
1. Créer une classe ***compteur*** permettant à tout moment de connaître le nombre d’objets existants.



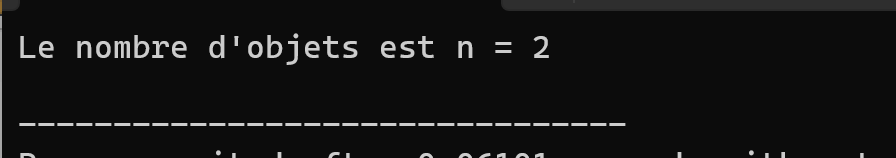
1. Ecrire une fonction essai dans laquelle, vous créez deux objets u et v de classe compteur.



1. Ecrire la fonction main dans laquelle, cous créez un objet a, vous appelez ensuite la fonction essai, puis vous créez un objet b.



4- Compiler et exécuter le programme. Conclure.



Apres compilation et exécution on remarque l’affichage du résultat qui 2 car les deux objets qui existent sont seulement a et b les deux autres (u et v) sont détruits.

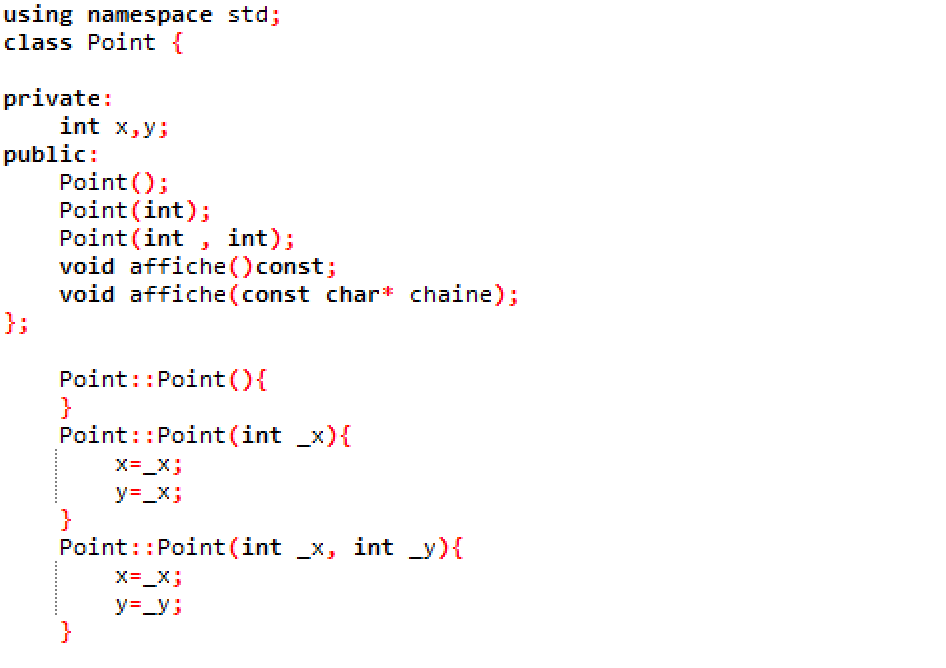
Exercice 2 :

1- Reprendre la classe ***point***, en utilisant la surdéfinition des fonctions membres pour créer plusieurs constructeur de cette classe.

- Premier constructeur : **point();** sans paramètres, il initialise x et y à 0,

- Premier constructeur : **point(int);** avec un seul paramètre, il initialise x et y à la valeur de ce paramètre,

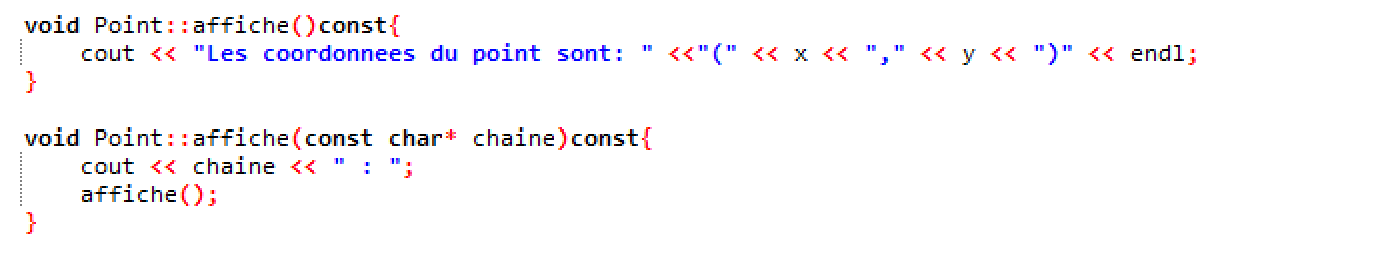
- Premier constructeur : **point(int, int);** avec deux paramètres, il initialise x à la valeur du premier paramètre et y à la valeur du second paramètre,



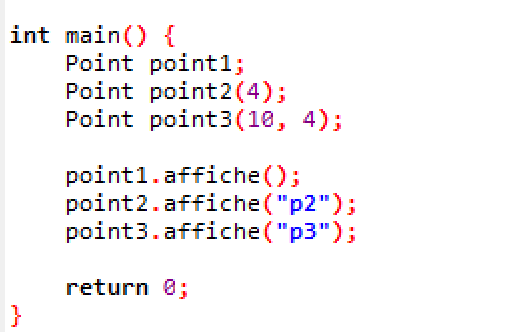
2- Ecrire les deux fonctions membres surchargées suivante :

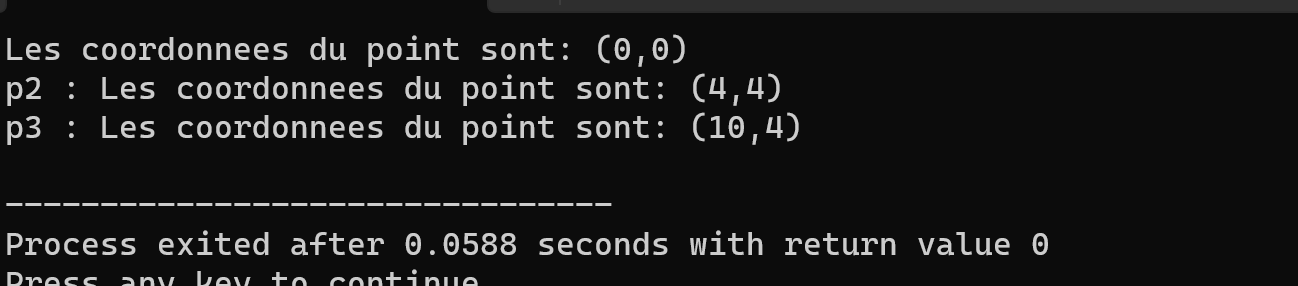
- Une fonction : **affiche();** sans paramètres, permettant d’afficher un point à l’écran,

- Une fonction : **affiche(char \*);** à un sel paramètre, une chaîne de caractère, permettant le nom du point et d’appeler la fonction : **affiche()ci dessus**.



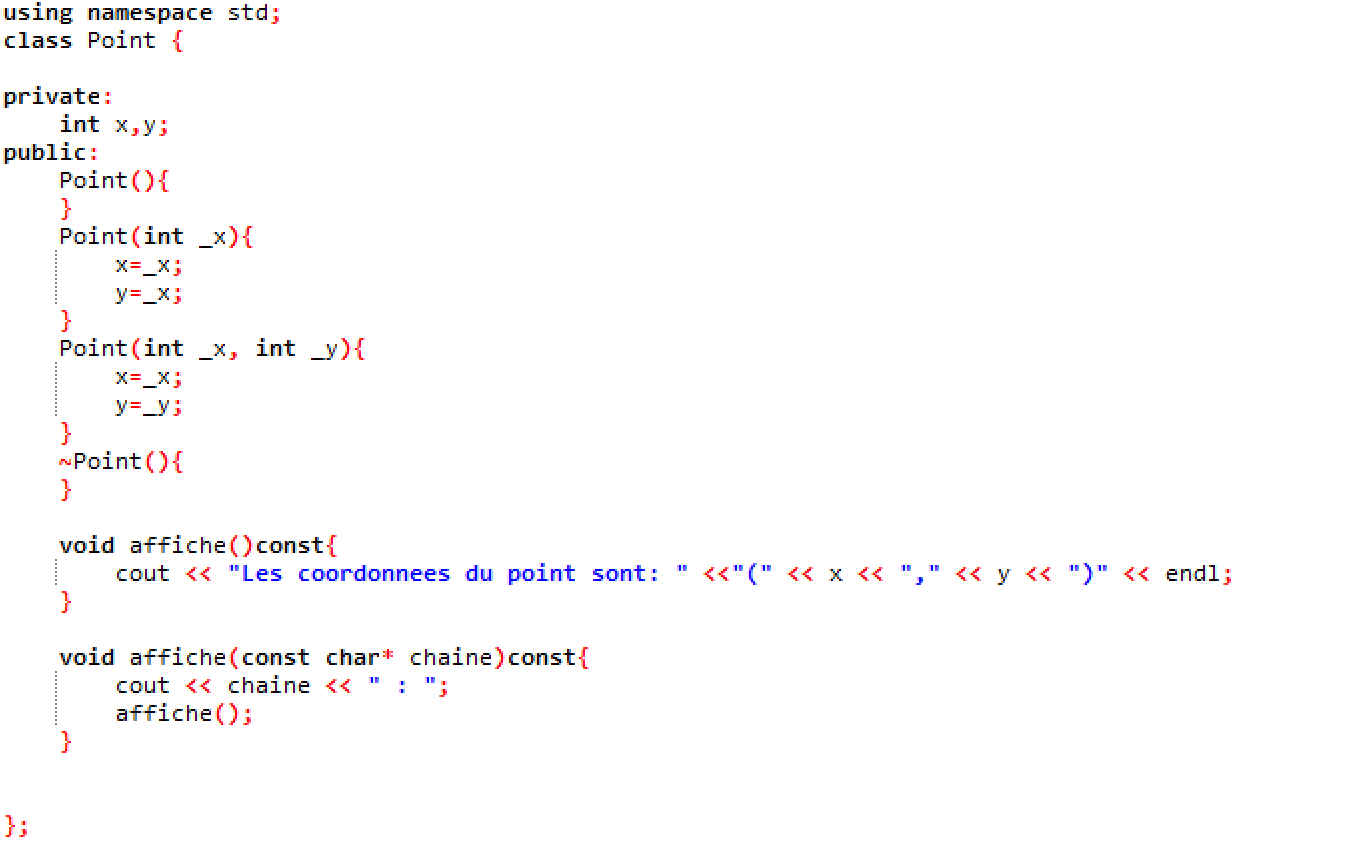
3- Proposer une fonction main() créant des objets de classe point et faisant appel à ces deux fonctions membres.





Exercice 3 :

1. Reprendre la classe ***point***, écrite dans l’exercice 2 ci-dessus en écrivant les différents constructeurs et fonctions membres sous forme de fonctions « ***inline*** ».



2- Comparer la taille du fichier objet « .obj » de cet exemple et celui de l’exemple précédent. Conclure ?

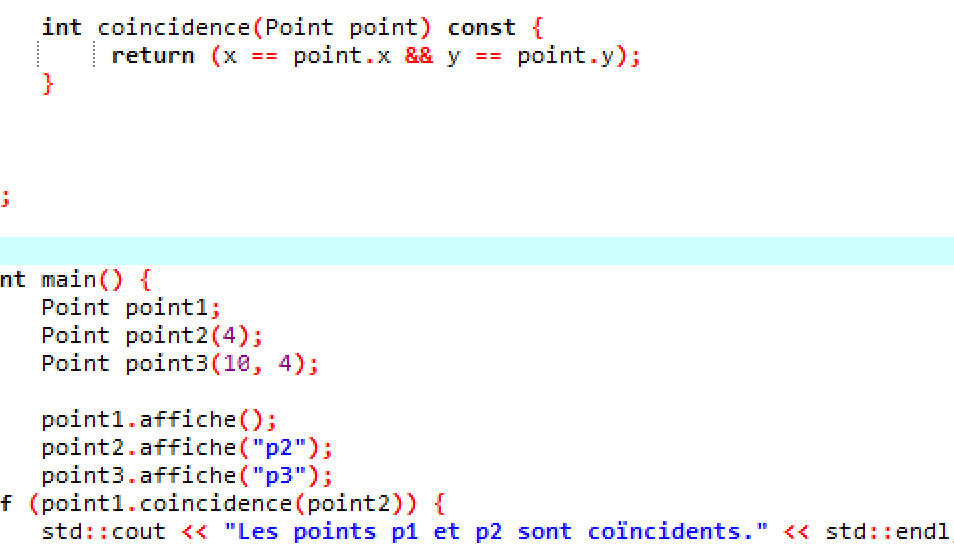




On remarque que le fichier contenant les fonctions en ligne est plus couteux en terme de mémoire .

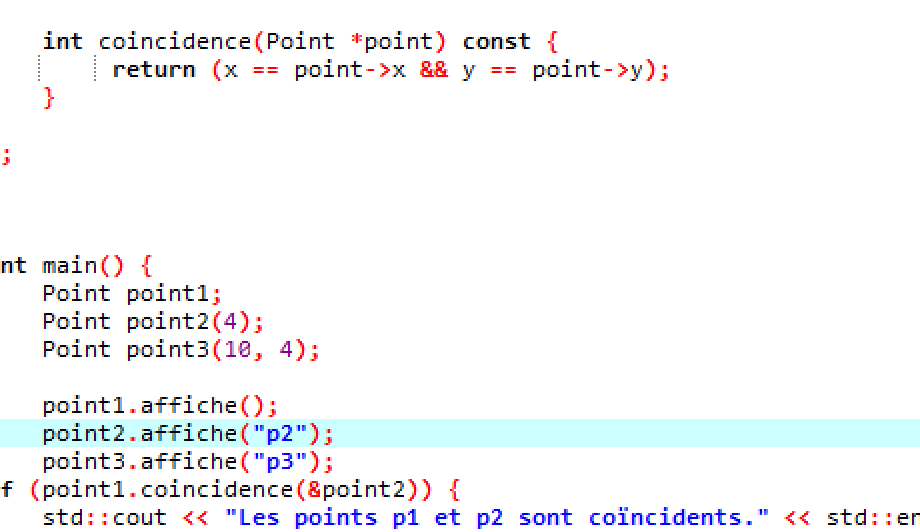
Exercice 4 :

Reprenons la classe ***point*** dans laquelle vous allez introduire une fonction membre nommée « ***coïncidence*** ». Cette fonction permet de détecter la coïncidence éventuelle entre deux points et qui a comme paramètre un objet de classe ***point***.

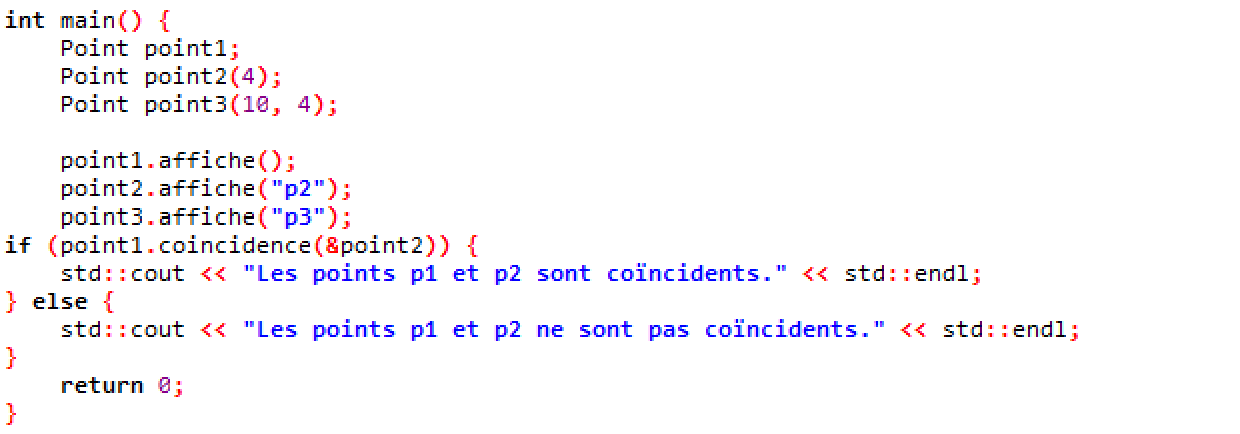


Exercice 5 : Passage de paramètres par adresse :

1. Modifier la fonction membre « ***coïncidence*** » de l’exemple précédent de sorte que son prototype devienne ***int point:: coïncidence (point \*adpt),***

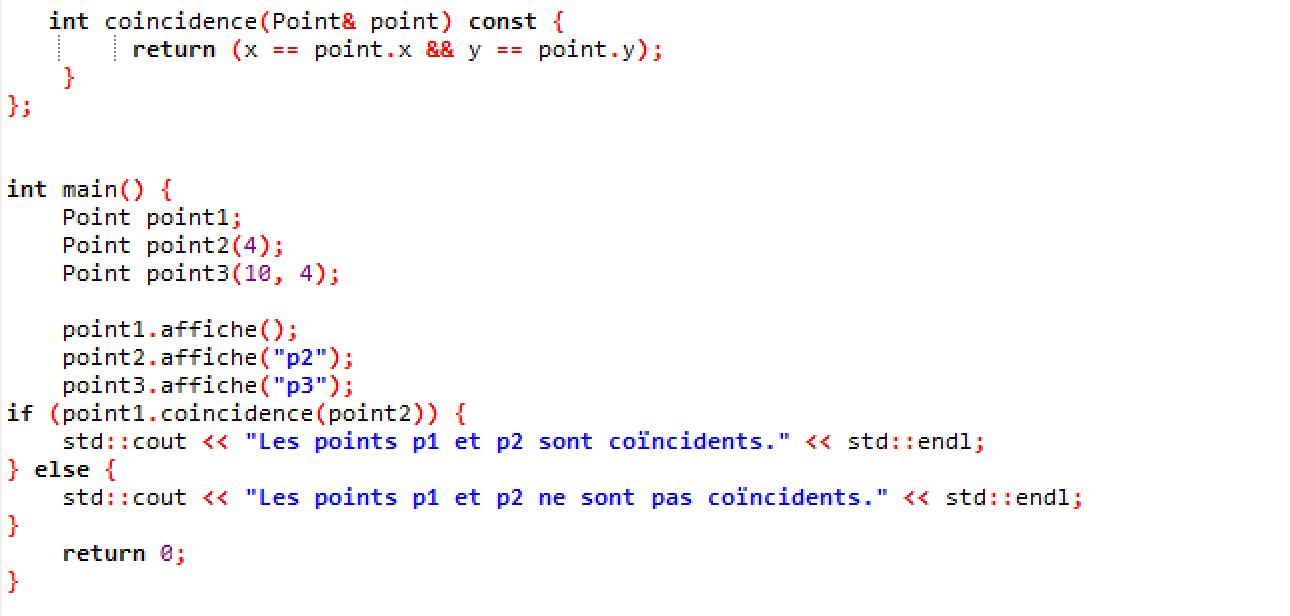


2- Ré-écrire le programme principal en conséquence.

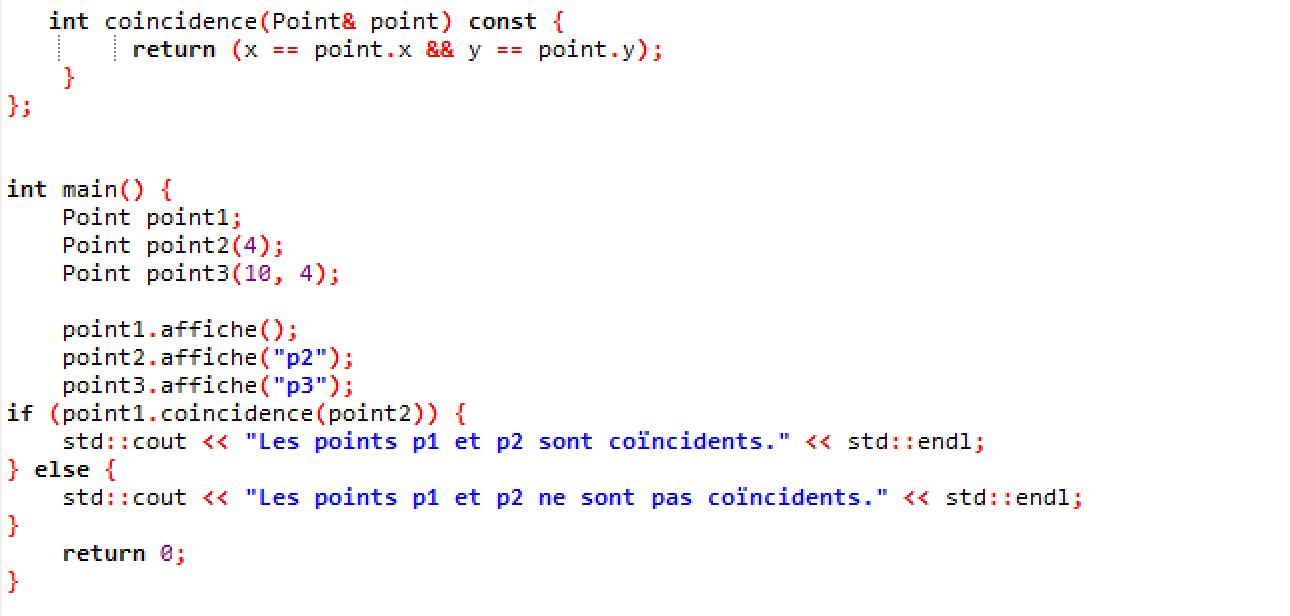


Exercice 6 : Passage de paramètres par référence :

1. Modifier la fonction membre « ***coïncidence*** » de l’exemple précédent de sorte que son prototype devienne ***int point:: coïncidence (point &adpt),***

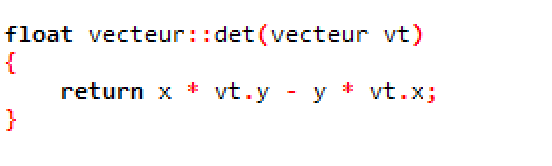


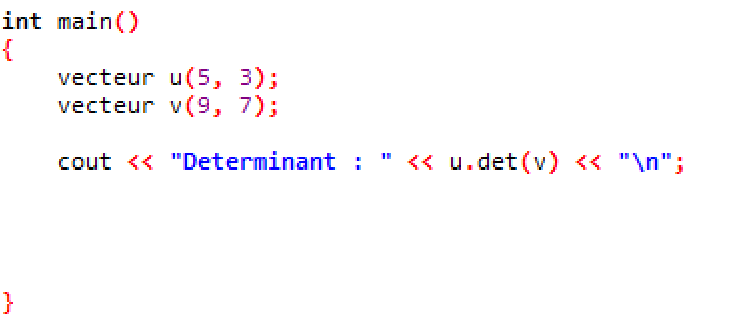
1. Ré-écrire le programme principal en conséquence.



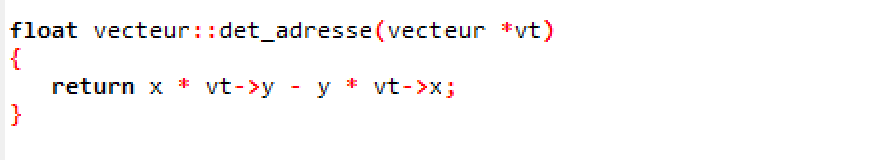
Exercice 7 :

1- Mettre cette classe en en oeuvre dans un programme principal ***void main()***, en ajoutant une fonction membre ***float det(vecteur)*** qui retourne le déterminant des deux vecteurs (celui passé en paramètre et celui de l’objet),

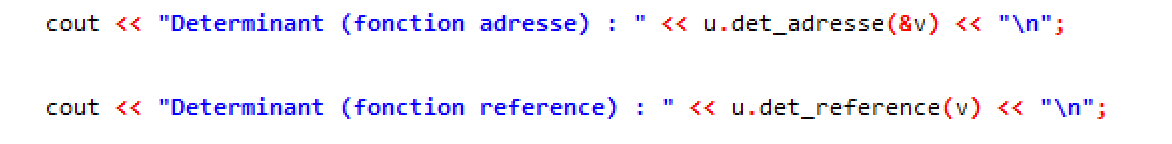




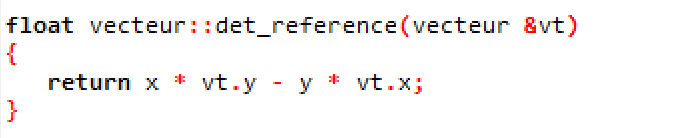
2- Modifier la fonction ***déterminant*** de sorte de passer le paramètre par adresse.



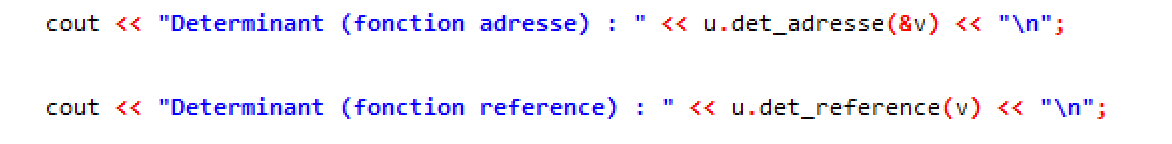
Appel :



3- Modifier la fonction déterminant de sorte de passer le paramètre par référence.



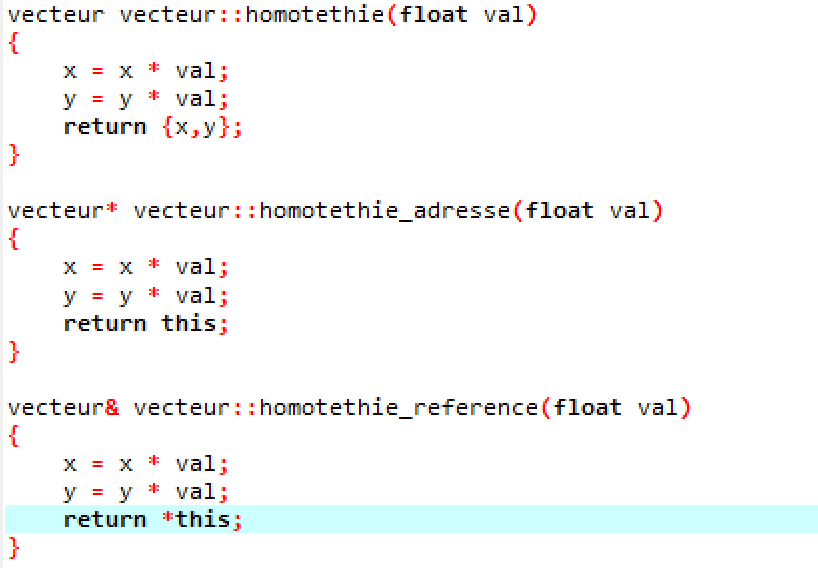
Appel :



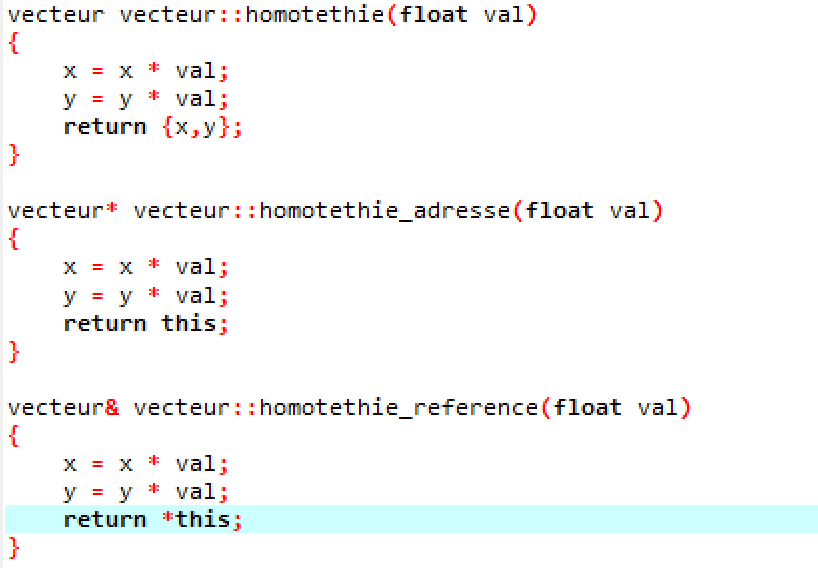
Exercice 8 : Classe vecteur 2 :

1- Modifier la fonction « homotéthie» qui retourne le vecteur modifié par valeur.

(prototype: vecteur vecteur::homotethie(float val)).



2- Modifier la fonction « homotéthie» qui retourne le vecteur modifié par adresse.



1. Modifier la fonction « homotéthie» qui retourne le vecteur modifié par référence.

