CY UNIVERSITÉ	**ING2 MF : PROGRAMMATION C++** **EXAMEN SESSION Normale** **DUREE : 1:30** **Aucun document n'est autorisé**		
Nom:	Prénom:	Groupe:	Créé le 06/12/2024

Exercice 1 : Soit le programme principal suivant:

```
void main() {
  int a = 1; int b = -2; int c = 0;
  ajouter(a,b,c);
  cout << "apres appel de ajouter" << endl;
  cout << "c = " << c << endl;
}
```

Questions :

1. Donner la sortie du programme dans les 3 cas suivants:

   a. **void** ajouter (**int** a, **int** b, **int** c) { c = a+b; }
   b. **void** ajouter (**int** a, **int** b, **int** * c) { *c = a+b; }
   c. **void** ajouter (**int** a, **int** b, **int** & c) { c = a+b; }

2. Comment on appelle chacun de ces 3 passages de paramètres ?

Exercice 2 : Soit la classe Forme est contenue dans le fichier nommée forme.h

```
#ifndef  FORME_H_
#define  FORME_H_
#include <iostream.h>

using namespace std;

class Forme {

public:

   virtual void description( const string s) {
    cout << "Ceci est une forme." << endl;
  }

   virtual const double aire(){};

  private:

    virtual double perimetre() const =0;
};

#endif
```

Questions :

1. Cette classe ne possède pas de constructeur, pourquoi ?
2. Que signifie le mot clé **virtual** dans chacune des 3 méthodes
3. Que signifie le mot clé **const** dans chacune les 3 méthodes
4. Pourquoi ce code ne compilera-t-il pas ? Expliquer

Exercice 3 : Soit la classe Point avec 2 attributs, position du point (x, y) , définie dans le fichier point.h

```
/////////////// fichier point.h ////////
#include <iostream.h>
using namespace std;

class Point {
private : short x, y;
public :
Point() { cout << " ++ normal \n"; }
~Point() { cout << " -- normal  \n"  ; }
} ;
```

Pour chacun des fichiers main.h suivants contenant la fonction main. Donner sa sortie et la commenter très brièvement

```
/////////////// fichier main.h  ////////////////
#include"point.h"
a. void main ()  {Point p;}
b. void main ()  {Point *p = new Point();}
c. void main ()  {Point *p = new Point(); delete p;}
```

Exercice 4 : On reprend la classe Point et on la modifie ainsi (voir ci-dessous).

```
//////////fichier point.h ////////////////////////////
#ifndef  POINT_H_
#define  POINT_H_

#include <iostream.h>
using namespace std;

class Point {
  private : short x, y;
  public :
     Point(short abs, short ord) : x(abs) , y(ord) {}
     virtual void decrire(){
        cout << "je suis un point \n";
        cout <<"mes coordonnees  : " << x << " " << y << "\n";
     }
     ~Point(){};
} ;

#endif"

//////////fichier pointcol.h ////////////////////
#include "point.h"

class PointCol : public Point{
  private
        unsigned int color;
  public :
    PointCol(short abs=0, short ord=0, unsigned int cl): Point(abs, ord){ color = cl;}
    void decrire (){
        cout << "je suis un point colore \n";
        cout <<"mes coordonnees  : " << x << " " << y
        cout << " et ma couleur : " << color <<"\n";
     }
};
```

```
////////////// fichier  main.h ////////////////////////////
#include "pointcol.h"

void main(){
        Point   *p1 = new Point(3,5);          p1-> decrire();
        PointCol  *pc1=new PointCol(8,6,2);    pc1-> decrire ();
        Point *p2=new PointCol(8,6,5);         p2-> decrire();
        PointCol  *pc2=new Point(7,6);         pc2-> decrire ();
}
```

Question :

1. Comment s'est faite l'initialisation des propriétés dans le constructeur? Et pourquoi?
2. Les propriétés sont définies avec le type **short** dans la classe Point et **unsigned int dans la classe PoinCol.** Quelle est la différence?
3. Cette classe Point est-elle une classe canonique ou pas commenter?
4. A quoi sert cette instruction "**using namespace** std" ? Si on le supprime, que faut-il ajouter pour que le programme compile
5. Expliquer la directive suivante "#**ifndef** POINT_H_ #**define** Point_H_ ….#**endif**"
6. Dans la déclaration de la classe fille pointCol, expliquer ce que signifie le spécificateur d'accès **public** devant la classe Point
7. Dans la classe fille nous avons un problème d'accès à ses propriétés, montrer ce problème? Comment le corriger?
8. Dans la fonction main(), comment appelle-t-on ce typage dans la construction de nos points.
9. Dans la fonction main une ligne est fausse, laquelle et pourquoi?
10. Supprimer cette ligne et afficher la sortie, le résultat.

Exercice 5 : On reprend les deux classes précédentes, *Point* et *PointCol* en corrigeant les erreurs, et on complète par deux autres classes *PointForme* qui contient un attribut de type chaine, pour dire comment le point sera dessiné (un point, un carré, un cercle, etc…), *et PointFormeCol*.

1. Donner le schema complet de votre programme avec toutes les classes, leurs attributs, leurs méthodes et leurs dépendances. (avec des boites comme UML)
2. Donner le fichier.h de chacune de ces 2 nouvelles classes ajoutées.
3. Dans la nouvelle méthode principale main(), définissez le template Vector<T> (un conteneur d'objets), ajouter un objet créé de chaque type à ce conteneur. Pour finir parcourir ce template et decrire chacun des objets contenus dans ce dernier en utilisant la method *decrire()* implémentée dans chacune des classes (utiliser un ierateur pour parcourir votre conteneur).

Annexe : méthodes de la classe template <typename ValueType> class Vector

Member functions
(constructor) Construct vector (public member function)
(destructor) Vector destructor (public member function)
operator= Assign content (public member function)

Iterators:
begin Return iterator to beginning (public member function)
end Return iterator to end (public member function)

Capacity:
size Return size (public member function)
max_size Return maximum size (public member function)
resize Change size (public member function)
capacity Return size of allocated storage capacity (public member function)
empty Test whether vector is empty (public member function)

Element access:
operator[] Access element (public member function)
front Access first element (public member function)
back Access last element (public member function)
data Access data (public member function)

Modifiers:
push_back Add element at the end (public member function)
pop_back Delete last element (public member function)
insert Insert elements (public member function)
erase Erase elements (public member function)
swap Swap content (public member function)
clear Clear content (public member function)