C++ pour les mathématiques appliquées

Les réponses doivent être concises. Aucun autre document autorisé.

Question 1.
Dans le contexte de la modélisation, pour chacun des éléments suivants, définissez-le, expliquez quand l'utiliser et donnez un exemple :

- Composition
- Agrégation

Question 2.
Décrivez le patron **Créateur** et son intérêt.

Décrivez brièvement les différents diagrammes utilisez lors de la conception logicielle, leur intérêt et leurs limites si elles existent.

Soit une classe `vecteur3d` définie par

```
1   class vecteur3d {
2     float x, y, z;
3     public :
4     vecteur3d (float c1=0.0, float c2=0.0, float c3=0.0)
5     {x = c1; y = c2; z = c3;}
6   };
```

Question 3.
Comment accéder à `x`, `y` et `z` ?

Question 4.
Introduire une fonction membre nommée coïncide permettant de savoir si deux vecteurs ont mêmes composantes :

- en utilisant une transmission par valeur,
- en utilisant une transmission par adresse,
- en utilisant une transmission par référence,

Question 5.
Comment accéder à une composante du vecteur depuis un programme extérieur et en lecture seule ?

Question 6.
Comment modifier le code afin de permettre la création de vecteurs d'entiers et de vecteurs de double ?

Question 7.
Donner le code de l'opérateur d'affectation (opérateur =) de la classe représentant un vecteur en dimension n

```
1     class vecteurXd
2     {
3       int n;
4       double* composante;
5     }
```

dans laquelle le tableau `val` est alloué dynamiquement.

Question 8.
Pour cette classe `vecteurXd`, est-il nécessaire de définir d'autres méthodes ? Si oui, lesquelles ? Pourquoi ?

Question 9.
Écrire une fonction qui retourne une référence au plus grand élément d'un `vecteurXd`.

Question 10.
On souhaite réaliser un outils de construction géométrique contenant des objets simples. Pour cela, on définit une classe `Forme` de la manière suivante

```
class Forme
{
  public :
  Forme(double cx, double cy): x(cx), y(cy){}
  // x et y sont les coordonnees du centre de l'objet.
  double x, y;
  virtual string type() = 0;
  virtual void info()
  {
    cout « endl « "Figure : " « type() « endl;
    cout « "Position : (" « x « "," « y « ")" « endl;
  }
}
```

Comment manipuler cet objet ?

Question 11.
On définit également une classe `Cercle` et une classe `Rectangle` de la manière suivante

```
class Cercle: public Forme
{
  public :
  Cercle(double px, double py, double r):
    Forme(px, py), rayon(r) { }
  virtual void info()
  {
    Forme::info();
    cout « "Rayon:" « rayon« endl;
  }

  virtual string type()
  {
    return "Cercle";
  }

}
class Rectangle : public Forme
{
  public :
  Rectangle(double px, double py, double w, double h):
    Forme(px, py), largeur(w), hauteur(h) { }

  virtual void info()
  {
    Forme::info();
    cout « "Largeur :" « largeur « endl;
    cout « "Heutaur :" « hauteur« endl;
  }

  virtual string type()
  {
    return "Rectangle";
  }

}
```

Le programme principal correspondant est

```
int main (int argc, char* argv)
{
  Forme **objs = new Forme[5];
  objs[0] = new Cercle(1,2,1);
  objs[1] = new Rectangle(-1,-2,2,1);
  objs[2] = new Rectangle(-2,1,1,3);
  objs[3] = new Cercle(1,-2,1);
  objs[4] = new Cercle(2,1,0.5);
  return 0;
}

```

Dans cette implémentation, il manque la destruction des différents objets.
Proposer une solution réalisant les différentes éléments de la destruction.

Question 12.
Expliquer le fonctionnement de la destruction dans une hiérarchie de classe.

Question 13.
Dans le code précédent, on souhaite remplacer le tableau par une liste afin de réaliser des insertions et suppressions dynamiques. Modifier la fonction `main` et décrire les propriétés du conteneur choisi.

Question 14.
Dans la STL, les conteneurs sont construits, entre autres, autour des itérateurs. Expliquer ce qu'est un itérateur et son intérêt en l'illustrant sur un exemple.