ÉPREUVE DE RATTRAPAGE DE LANGAGE C++

à préparer à l'avance en vue d'une présentation orale de 20 minutes

---

Nous nous proposons de définir une classe définissant les matrices mathématiques. Pour ce faire, nous définissons une classe de base `Object` élémentaire et la classe dérivée `Matrix` qui définit une matrice de doubles. Ces classes sont déclarées dans les fichiers d'en-tête (`.h`) comme suit :

```
class Object {
public :
  virtual void display() const ;
  virtual Object* clone() const=0 ;
} ;

class Matrix : public Object
{
public :
  double& at(int line,int col) ;

  Matrix(int numLines,int numCols) ;
  Matrix(const Matrix& m) ;
  ~Matrix() ;

  bool operator==(Matrix m) const ;
  bool operator!=(Matrix m) const ;
  Matrix operator=(const Matrix& m) ;

  friend Matrix operator+(const Matrix& m1,const Matrix& m2) ;
  friend Matrix operator+(const Matrix& m1,const double* m2) ;
  friend Matrix operator+(const double* m1, const Matrix& m2) ;

  void display() const ;
  Object* clone() const=0 ;
protected :
  double* array ;
  int numLines, numCols ;
} ;
```

Vous fournirez, au fil de vos réponses aux questions ci-dessous, l'implémentation des classes `Object` et `Matrix`.

La classe `Matrix` utilise le tableau à **une** dimension `array` afin de représenter le contenu d'une matrice, et les deux données membres `numLines` et `numCols` pour préciser les dimensions de la matrice. L'indice `i` de l'élément du tableau `array` correspond alors à la ligne `l` et la colonne `c` de la matrice selon la formule `i = (l * numCols) + c`.

Questions

1. Retour d'une référence : La fonction membre `at` retourne une référence sur le double situé à la ligne `line` et la colonne `col` de la matrice.

   (a) En quoi est-il intéressant que la fonction `at` retourne une référence ?

   (b) Donner une implantation possible de cette fonction.

   (c) Pourquoi n'utilise-t-on pas l'opérateur l'indexation au lieu de la fonction `at` ?

   Remarque – Il sera pertinent d'utiliser la fonction `at` dans l'implantation de la classe `Matrix` demandée dans les questions suivantes.

2. Constructeurs :

   (a) Dans le code `Matrix* m = new Matrix(2,3)`, est-ce le constructeur qui réalise l'allocation mémoire de `m`, l'opérateur `new`, ou les deux (justifier votre réponse) ?

   (b) Donner une implantation possible du constructeur `Matrix(int numLines,int numCols)`.

3. Constructeur de copie :

   (a) Quel est le rôle d'un constructeur de copie ?

   (b) Par quelles fonctions ou/et opérateurs de `Matrix` ce constructeur est-il automatiquement (implicitement) invoqué ?

   (c) Donner une implantation possible de ce constructeur.

4. Destructeur :

   (a) Le destructeur `~Matrix` est-il utile ici ? Justifier votre réponse.

   (b) Si oui, donner une implantation possible de ce destructeur ; si non, expliquer en quoi il n'est pas utile ici.

5. Opérateurs de test d'égalité et d'inégalité :

   (a) À quoi sert le mot clé `const` positionné à la fin des deux opérateurs de test d'égalité et d'inégalité ?

   (b) Quel est l'intérêt de passer la matrice `other` par référence ?

   (c) Pourquoi alors le mot clé `const` est-il positionné devant cet argument (`const Matrix& m`) ?

(d) Donner une implantation possible des opérateurs de test d'égalité et d'inégalité.

6. Opérateur d'affectation :
   (a) L'opérateur d'affectation existe par défaut pour chaque classe. Que réalise cette version par défaut et pourquoi doit-il alors être redéfini ici ?
   (b) Pourquoi l'opérateur d'affectation est-il déclaré comme renvoyant une matrice ?
   (c) Donner une implantation possible de cet opérateur.
7. Opérateurs amis d'addition de matrices :
   Les deux dernières surcharges de l'opérateur d'addition permettent la manipulation d'instances de `Matrix` avec des tableaux C de `doubles`. Exemple :

```
Matrix a(2,3) ;
const double* b = {
  1.0, 2.0, 3.0 ,
  4.0, 5.0, 6.0 }
Matrix c = a + b ;
```

   A l'issue de ce code, la matrice `c` possède la même valeur que la matrice représentée par le tableau `b`.
   (a) Qu'est-ce qui justifie l'utilisation du mot clé `friend` pour cette utilisation de l'opérateur + ?
   (b) Donner une implantation possible de `Matrix operator+(const Matrix& m1,const double* m2)`.
   (c) Donner une implantation possible de `Matrix operator+(const Matrix& m1,const Matrix& m2)`.
8. Fonction membre virtuelle `display` (elle réalise l'affichage textuel de la matrice) :
   (a) Quel mécanisme typique des langages à objets le mot clé `virtual` permet-il d'enclencher et à quoi sert-il ?
   (b) D'après sa déclaration donnée dans l'énoncé, la classe `Object` doit-elle fournir le code de la fonction membre `display` ? Justifier votre réponse.
   (c) Si oui, proposer une implantation de cette méthode pour la classe `Object`.
   (d) D'après sa déclaration donnée dans l'énoncé, la classe `Matrix` doit-elle fournir le code de la fonction membre `display` ? Justifier votre réponse.
   (e) Si oui, proposer une implantation de cette méthode pour la classe `Matrix`.
9. Fonction membre virtuelle `clone` (elle renvoie une copie de l'objet cloné) :
   (a) D'après sa déclaration donnée dans l'énoncé, la classe `Object` doit-elle fournir le code de la fonction membre `clone` ? Justifier votre réponse.
   (b) Si oui, donner une implantation possible de la méthode `clone` da la classe `Object`.
   (c) D'après la déclaration donnée dans l'énoncé, la classe `Matrix` doit-elle fournir le code de la fonction membre `clone` ? Justifier votre réponse.
   (d) Si oui, donner une implantation possible de la méthode `clone` da la classe `Matrix`.