POLYTECH – ELEC4 2021–2022

Contrôle de Programmation C++

Durée : 2h Aucun document autorisé

Nom : Prénom :

1 La classe abstraite matrice

▶ 1. Écrivez la classe abstraite *générique* `matrice`. On appellera `T` le type générique des éléments de la matrice. Cette classe contient :

- 3 variables membres protégées, `nbElem`, `nbL` et `nbC`, qui sont respectivement le nombre d'éléments contenus dans la matrice, son nombre de lignes et de colonnes ;
- un constructeur public avec 2 paramètres (le nombre de lignes et de colonnes, par défaut égaux à 0) pour initialiser les 3 variables membres ;
- un constructeur de copie public ;
- le destructeur public (déclaré **virtual**) ;
- la méthode publique virtuelle pure `get` qui renvoie l'élément de type `T` en (i, j) ;
- la méthode publique virtuelle pure `set` qui met un l'élément `x` de type `T` en (i, j) ;
- la méthode publique virtuelle pure `toString` qui renvoie la représentation sous forme de `std::string` de l'objet courant ;
- la surcharge publique de l'opérateur `<<` pour écrire une `matrice<T>` sur un `std::ostream` ;

2 La classe matriceDouble

On souhaite manipuler des matrices de réels **double**. Pour cela, vous allez définir la classe `matriceDouble` qui hérite de la classe générique `matrice` précédente, en fixant le type des éléments au type **double**. Les réels **double** de la `matriceDouble` seront placés dans un tableau à une dimension appelé `mat` et alloué **dynamiquement**,

▶ 2. Écrivez la classe `matriceDouble` avec :
- la variable protégée `mat` de type **double** `*` (je vous aide) ;
- un constructeur public pour initialiser une `matriceDouble` $m \times n$ à une valeur `v` (par défaut 0.0) ;
- le constructeur public de copie ;
- le destructeur public ;
- l'implémentation des méthodes virtuelles `get`, `set` (vérifiez la validité des indices) et `toString`.
- la surcharge publique de l'opérateur d'affectation ;
- la surcharge publique de l'opérateur `<<`, si c'est utile. Oui ou non, vous expliquez.

► 3. Est-il possible d'écrire dans la classe `matrice` la surcharge publique de l'opérateur `()` pour accéder à un élément d'une `matriceDouble` ? Si on a une `matriceDouble m`, on pourra par exemple écrire `m(2,1)` pour accéder au réel double de la 3ème ligne et de la 2ème colonne. Si c'est possible (expliquez), écrivez cette surcharge.

► 4. Dans la classe `matrice`, ajoutez la déclaration de la méthode publique virtuelle pure `subMat` qui possède 4 paramètres entiers `i1`, `i2`, `j1` et `j2` (avec `i1`⩽`i2` et `j1`⩽`j2`), et qui renvoie un pointeur sur un objet de type `matrice<T>` formé des lignes `i1` à `i2` et des colonnes `j1` à `j2`.

► 5. Dans la classe `matriceDouble`, écrivez l'implémentation de la méthode publique `submat` (sans oublier son en-tête). Si les indices ne sont pas valides, vous émettrez l'exception `IndexInvalide`. Vous déclarerez la classe `IndexInvalide` héritière de `std::exception`. L'exception indiquerera le ou les indices invalides.

▶ 6. Écrivez la fonction `main` qui :

- déclare une `matricedDouble m1` 3×6 dont les éléments sont initialisés à 1.0 ;
- écrit sur la sortie standard `m1`
- déclare `matricedDouble m2` initialisée à partir de `m1` ;
- crée et écrit sur la sortie standard la sous-matrice prise entre les lignes 1 et 2 et les colonnes 1 et 3 de `m2`. Vous gérerez l'exception.