Programmation C++ Avancée

Joel Falcou Guillaume Melquiond

5 décembre 2017

Le fichier source obtenu à la fin de ce TP est à envoyer à `joel.falcou@lri.fr` et `guillaume.melquiond@inria.fr`. Vous insérerez vos réponses (courtes) aux différentes questions sous forme de commentaires dans le fichier source.

1 Matrices denses

Implémentez une classe pour représenter des matrices (tableaux bidimensionnels de `double`). Cette classe devra fournir une méthode `size` renvoyant une paire hauteur/largeur et une méthode `get` prenant des numéros de ligne et de colonne et renvoyant le contenu de la case correspondante (lignes et colonnes seront numérotées à partir de zéro).

```
class full_matrix {
 public:
  full_matrix(int height, int width, std::vector<double> const &v);
  std::pair<int,int> size() const;
  double get(int i, int j) const;
};
```

Le constructeur prendra en argument les dimensions de la matrice ainsi qu'un vecteur qui servira à initialiser les cases de la matrice (ligne par ligne). Si le vecteur n'est pas assez long, les valeurs manquantes seront initialisées à zéro.

Définissez une fonction permettant d'afficher le contenu d'une matrice :

```
std::ostream &operator<<(std::ostream &out, full_matrix const &m);
```

Testez votre code avec la fonction `main` suivante :

```
int main() {
  full_matrix m(2, 4, {1., 2., 3., 4., 5., 6.});
  std::cout << m << '\n';
  return 0;
}
```

Le choix de l'affichage est libre, du moment que la distinction entre ligne et colonne est claire. Voici une possibilité pour l'exemple ci-dessus : `[1 2 3 4; 5 6 0 0]`.

Ajoutez un constructeur permettant d'éviter les copies inutiles quand c'est possible :

```
full_matrix(int h, int w, std::vector<double> &&v);
```

2 Gestion des erreurs

Faites que la méthode `get` lève une exception en cas d'accès à une case en dehors de la matrice.

Indiquez pour chaque méthode de `full_matrix` sa garantie d'exception. Utilisez le mot-clé `noexcept` si elle ne peut pas lever d'exception. Sinon indiquez en commentaire `basic` ou `strong` ainsi qu'un exemple de problème qui peut survenir.

Ce travail de gestion/documentation des erreurs devra aussi être effectué dans toute la suite du TP.

3 Matrices génériques

Certaines matrices (diagonales, triangulaires, etc) contiennent de nombreux zéros qu'il est inutile de stocker ; la classe `full_matrix` n'est alors plus adaptée. La classe abstraite suivante sera donc utilisée à la place :

```
class matrix {
 public:
  virtual std::pair<int,int> size() const = 0;
  virtual double get(int i, int j) const = 0;
  virtual ~matrix() = default;
};
```

Faites hériter `full_matrix` de `matrix`. Modifiez la signature de `operator<<` pour qu'il se contente d'un argument de type `matrix`. Assurez vous que la fonction `main` ci-dessus fonctionne toujours.

Définissez une classe `diag_matrix` représentant les matrices diagonales. Le vecteur passé en argument du constructeur servira à initialiser les éléments de la diagonale. Si ce vecteur a n éléments, la matrice aura une taille $n \times n$. Vous prendrez soin d'écrire la classe de telle sorte que l'espace mémoire occupé corresponde à n nombres de type `double` et non pas n^2.

```
class diag_matrix: public matrix {
 public:
  diag_matrix(std::vector<double> const &v);
};
```

Ajoutez les lignes suivantes à `main` pour vous assurer que votre code fonctionne correctement.

```
std::shared_ptr<matrix> d(new diag_matrix({7., 8., 9., 10.}));
std::cout << *d << '\n';
```

4 Sous-matrices

Ajoutez à `matrix` une méthode ayant la signature suivante :

```
std::shared_ptr<matrix> slice(int i, int j, int h, int w) const;
```

La matrice renvoyée correspondra à la sous-matrice de `this` commençant à la ligne i et à la colonne j et ayant h lignes et w colonnes. Remarque : cette nouvelle matrice devra utiliser aussi peu de mémoire que possible, c'est-à-dire qu'elle ne devra pas dupliquer les valeurs stockées dans `this`.

Testez votre code en ajoutant les lignes suivantes à `main` :

```
std::shared_ptr<matrix> s = d->slice(0, 1, 3, 2)->slice(1, 0, 2, 1);
std::cout << *s << '\n'; // affichage attendu : [8; 0]
```

Expliquez pourquoi prendre la sous-matrice d'une sous-matrice n'est pas aussi efficace qu'on pourrait l'espérer. Modifiez votre code pour corriger ce défaut.