



POA - Labo 1

Matrix Reloaded

Nicolas Crausaz & Maxime Scharwath

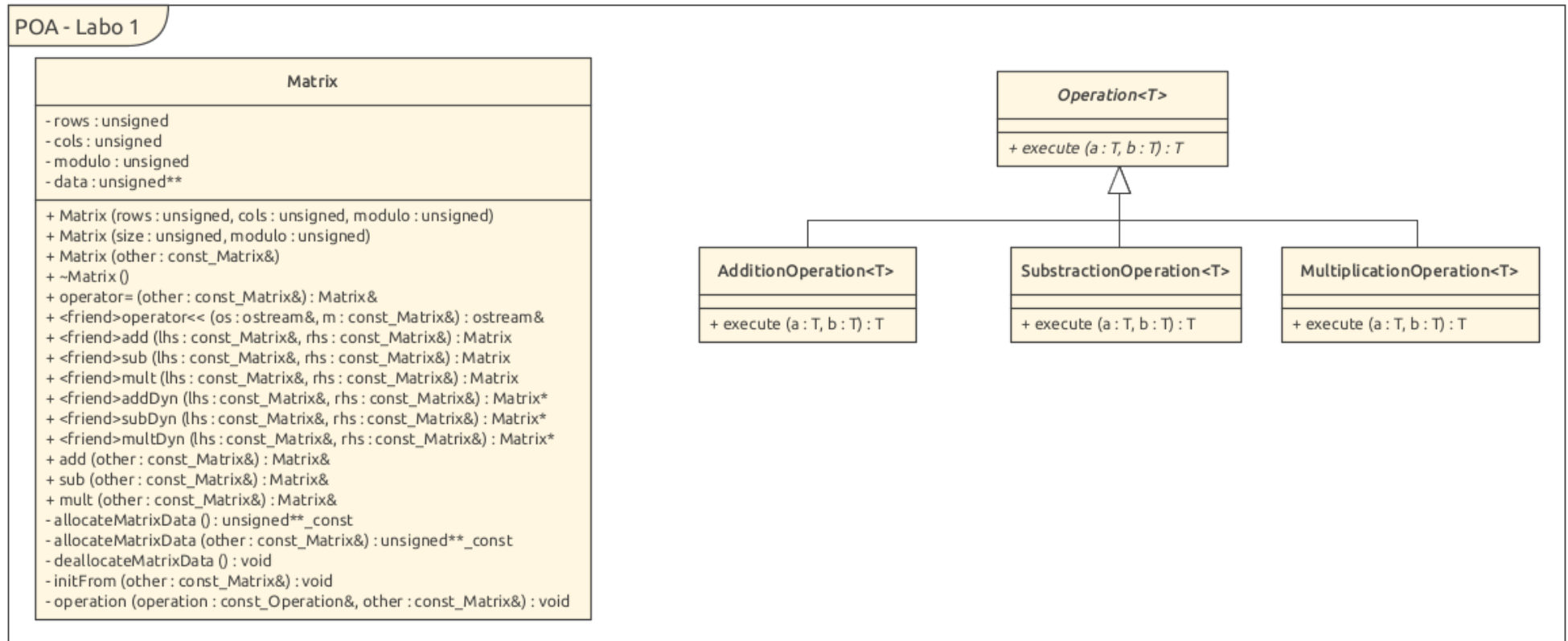
14.03.2022

Table des matières

1. Conception	3
1.1. Diagramme	3
2. Implémentation	4
2.1. Choix	4
Classe Matrix	4
Classes Operation	4
3. Tests	5
3.1. Constructeurs	5
3.1. Opérations	6

1. Conception

1.1. Diagramme



2. Implémentation

2.1. Choix

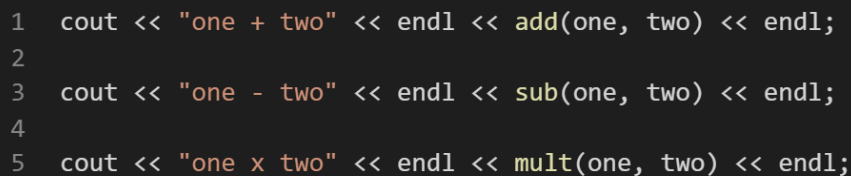
Classe Matrix

Nous avons ajouté un constructeur supplémentaire pour permettre la création de matrices carrées directement. Il suffit d'y passer la taille et le modulo en paramètre. Ce constructeur spécifique appelle le constructeur de matrice lignes colonnes.

Les opérations sur les matrices ont été, comme demandé, implémentées de trois manières différentes:

1. Modification de la matrice sur laquelle la méthode est invoquée
2. Création d'une nouvelle matrice par valeur
3. Création d'une nouvelle matrice dynamiquement (retourne pointeur)

Nous avons donc décidé d'implémenter les méthodes 2 et 3 en tant que fonctions friend, en effet ces dernières ne modifiant pas l'instance d'une matrice (création d'une nouvelle). Ces deux fonctions prennent ainsi deux matrices en paramètres. Exemple:



```
1  cout << "one + two" << endl << add(one, two) << endl;  
2  
3  cout << "one - two" << endl << sub(one, two) << endl;  
4  
5  cout << "one x two" << endl << mult(one, two) << endl;
```

Classes Operation

La classe *Operation* générique et abstraite, elle propose uniquement la méthode abstraite `execute()`. Nous avons décidé d'implémenter cette classe (et sous-classes) de manière générique étant donné que leur but est d'effectuer des opérations basiques sur des types numériques variés (int, unsigned, float, double,...). Dans le cadre de ce laboratoire, les matrices n'utilisent que des opérations sur des unsigned. Ce choix d'implémentation ne complexifie pas l'utilisation de ces classes et permet une meilleure flexibilité si l'on réutilisait les opérations pour un autre but dans un logiciel plus complet.

Dans l'exemple ci-dessous, nous avons choisi de mettre la création d'opération en *static* pour éviter la création/destruction de cet objet à chaque fois que l'on effectue une opération. En plus avec notre conception, les 3 méthodes d'opération (dynamique, copie, et modification de l'instance) utilisent cette méthode en interne.

```

1  Matrix& Matrix::add(const Matrix& other) {
2      static AdditionOperation<DataType> op;
3      operation(op, other);
4      return *this;
5  }
```

Puisque nous utilisons des entiers signés, l'opération de soustraction va fonctionner différemment: par exemple: $3-6=-3$ mais en signé $3-6=4294967293$ mais puisque nous effectuons un modulo sur ce résultat, la valeur obtenue revient à une valeur cohérente et qui suivent la logique de l'exemple donné dans la donnée du laboratoire.

3. Tests

Les tests ci-dessous sont numérotés et implémentés dans la fonction *unit_tests()* du programme principal. Les tests suivants sont vérifiés en affichant les matrices grâce à l'opérateur << implémenté et testé par nos soins. sauf indication contraire, le modulo utilisé est 8.

3.1. Constructeurs

Numéro du test	Description	Résultat attendu (exemple)	Test passé
1a	Matrice avec modulo 0	exception runtime_error	ok
1b	Matrice carrée avec mod. 0	exception runtime_error	ok
2a	Matrice avec nombre de lignes et colonnes invalides	exception runtime_error	ok
2b	Matrice avec nombre de colonnes invalide	exception runtime_error	ok
2c	Matrice avec nombre de lignes invalide	exception runtime_error	ok
2d	Matrice carrée avec taille invalide	exception runtime_error	ok

3a	Matrice valide	2 3 1 2 0 4 7 3 5 5 5 2 0 6 1 0 2 1 5 7	ok
3b	Matrice carrée valide	6 4 6 2 3 2 5 4 7 4 7 2 0 0 5 0	ok
4a	Matrice avec 1 ligne	1 4 3 5	ok
4b	Matrice avec 1 colonne	4 2 0 1	ok
5a	Construction par copie	6 1 0 3 6 1 0 3 2 6 7 4 2 6 7 4 7 1 1 3 7 1 1 3	ok
5b	Affectation	7 1 0 0 1 5 4 2 4 7 1 0 0 1 5 4 2 4	ok

3.1. Opérations

Numéro du test	Description	Résultat attendu (exemple)	Test passé
6a	Addition matrices de même tailles	7 4 4 4 6 4 1 6 1 0 7 4 3 1 4 4 6 1 0 2 + 2 3 0 1 3 6 1 3 0 4 4 7 0 1 3 7 6 4 5 7 = 1 7 4 5 1 2 2 1 1 4 3 3 3 2 7	ok

		3 4 5 5 1	
6b	Addition matrices de tailles différentes	3 4 2 1 6 1 6 4 + 7 5 4 7 3 0 = 2 1 2 1 2 0 6 4 3 0 0 0	ok
6c	Addition (copie) matrices de même tailles	7 7 1 7 4 7 4 5 5 5 0 2 1 4 7 7 4 7 0 7 + 7 6 6 7 0 3 5 1 4 2 5 4 1 7 3 5 6 0 3 3 = 6 5 7 6 4 2 1 6 1 7 5 6 2 3 2 4 2 7 3 2	ok
6d	Addition (copie) matrices de tailles différentes	7 5 2 0 1 7 1 3 + 7 3 7 5 2 0 = 6 0 2 0 0 4 1 3 2 0 0 0	ok
6e	Addition (dynamique) matrices de même tailles	6 5 0 3 7 6 7 0 4 4 3 6 6 0 0 4 4 7 7 6 + 7 7 4 7 3 0 4 5 6 6 2 5 3 2 0 2 0 0 3 4 = 5 4 4 2 2 6 3 5 2 2	ok

		5 3 1 2 0 6 4 7 2 2	
6f	Addition (dynamique) matrices de tailles différentes	7 6 0 3 4 0 4 2 + 0 2 2 1 3 3 = 7 0 0 3 6 1 4 2 3 3 0 0	ok
7a	Soustraction matrices de même tailles	7 7 0 0 7 6 2 6 7 6 2 6 2 5 2 5 1 1 1 5 - 4 5 1 5 3 3 6 7 1 5 3 1 5 4 1 4 3 4 2 3 = 3 2 7 3 4 3 4 7 6 1 7 5 5 1 1 1 6 5 7 2	ok
7b	Soustraction matrices de tailles différentes	3 5 1 5 2 4 3 4 - 5 4 1 1 2 3 = 6 1 1 5 1 3 3 4 6 5 0 0	ok
7c	Soustraction (copie) matrices de même tailles	7 6 7 5 5 1 3 1 2 0 6 3 5 1 7 0 4 2 6 6 - 0 1 3 3 5 0 0 7 2 3 3 1 1 2 7 7 3 3 0 5 = 7 5 4 2 0 1 3 2 0 5 3 2 4 7 0	ok

		1 1 7 6 1	
7d	Soustraction (copie) matrices de tailles différentes	4 7 0 2 0 0 3 5 - 3 1 4 3 3 7 = 1 6 0 2 4 5 3 5 5 1 0 0	ok
7e	Soustraction (dynamique) matrices de même tailles	7 1 0 7 1 3 3 4 5 4 6 5 4 2 0 5 7 5 4 0 - 7 5 1 3 2 4 4 7 7 0 7 7 2 0 7 3 3 2 7 0 = 0 4 7 4 7 7 7 5 6 4 7 6 2 2 1 2 4 3 5 0	ok
7f	Soustraction (dynamique) matrices de tailles différentes	7 6 5 3 0 6 0 0 - 4 4 1 3 1 3 = 3 2 5 3 7 3 0 0 7 5 0 0	ok
8a	Multiplication matrices de même tailles	6 4 7 3 3 7 4 3 6 6 3 6 2 6 0 1 7 7 0 5 * 3 1 4 3 2 0 0 4 4 1 7 3 5 7 7 1 6 3 5 5 = 2 4 4 1 6 0 0 4 0 6 5 2 2 2 0 1 2 5 0 1	ok

8b	Multiplication matrices de tailles différentes	2 0 4 4 7 4 6 6 4 6 4 0 7 0 3 1 0 3 6 4 * 5 5 7 3 4 6 5 3 2 2 1 4 2 5 1 1 2 7 0 7 = 2 0 4 4 4 0 6 2 0 4 4 0 6 0 3 1 0 5 0 4	ok
8c	Multiplication (copie) matrices de même tailles	5 4 7 5 5 3 7 5 6 5 2 4 2 2 7 7 1 4 2 3 * 6 4 0 1 2 1 2 4 0 3 4 6 0 3 3 5 6 2 3 5 = 6 0 0 5 2 3 6 4 0 7 0 0 0 6 5 3 6 0 6 7	ok
8d	Multiplication (copie) matrices de tailles différentes	7 5 1 2 0 0 1 1 * 5 4 5 3 0 5 = 3 4 0 0 0 0 0 0 0 0 0 0	ok
8e	Multiplication (dynamique) matrices de même tailles	5 3 7 7 0 7 3 4 5 3 0 0 1 6 3 5 4 2 2 6 * 4 3 6 6 4 4 2 1 0 3 7 5 6 6 5 6 6 0 2 3 = 4 1 2 2 0 4 6 4 0 1	ok

		0 0 6 4 7 6 0 0 4 2	
8f	Multiplication (dynamique) matrices de tailles différentes	7 1 3 5 2 7 6 6 * 1 5 6 6 2 6 = 7 5 0 0 4 2 0 0 0 0 0 0	ok