Laboratoire 1: Matrix Reloaded

Durée du laboratoire: 5 périodes. Rendre le code source (.cpp et .hpp uniquement) le jeudi 17 mars au début de la séance de laboratoire sur Cyberlearn.

Définir une classe permettant de représenter des matrices de taille quelconque (NxM) contenant des éléments entre 0 et n − 1 pour un entier n (les entiers sont modulo n) qui réponde aux contraintes ci-dessous.

- Il soit possible de créer une matrice soit en générant son contenu aléatoirement (une fois sa taille et son modulo connus).
- Stocker les valeurs de la matrice dans un tableau de tableaux (ne pas utiliser de conteneur C/C++).
- Il soit possible d'afficher le contenu de la matrice en utilisant l'opérateur d'écriture dans un flux (`<<`).
- Il soit possible de dupliquer une matrice.
- Il soit possible d'effectuer les opérations suivantes entre deux matrices: l'addition, la soustraction et le produit composante par composante. Toutes les opérations doivent être effectuées modulo n.
  Le résultat C d'une multiplication composante par composante entre une matrice A et une matrice B est défini par $C_{i,j} = A_{i,j} \cdot B_{i,j} \bmod n$.
- Chaque opération peut être effectuée de trois manières différentes:
  - Soit en modifiant la matrice sur laquelle est invoquée la méthode,
  - Soit en retournant, par valeur (pourquoi par valeur et non par référence?) une nouvelle matrice résultat allouée statiquement,
  - Soit en retournant, un pointeur sur une nouvelle matrice résultat allouée dynamiquement.
- Si l'on effectue une opération entre une matrice A de taille $M_1 \times N_1$ et une matrice B de taille $M_2 \times N_2$ et que les tailles ne correspondent pas, le résultat est une matrice de taille $\max(M_1, M_2) \times \max(N_1, N_2)$ où les $A_{i,j}$ et $B_{i,j}$ manquants ont été remplacés par des 0.
- Si les modules n des deux matrices ne correspondent pas, lever une exception de type `invalid_argument` (voir `<except>`).
- En cas de toute autre erreur, lever une exception de type `runtime_error`.
- Ne pas utiliser d'expression lambda.

Remarques

- Les fonctions `srand`, `time`, `swap` (voir `<cstdlib>`, `<ctime>` et `<algorithm>`) seront sûrement utiles.
- Attention au comportement de l'opérateur `%` en C++, en particulier pour la soustraction.
- Prendre garde à ce que toute la mémoire allouée dynamiquement soit libérée correctement (allocation dans un constructeur => définition du destructeur, du constructeur de copie, et de l'opérateur `=`).
- Pour qu'une méthode supporte la liaison dynamique il faut qu'elle soit déclarée *virtuelle* (mot clef `virtual`).
- Prendre garde à ce que la construction d'une matrice résultat s'effectue correctement.
- Génération d'un nombre aléatoire entre 1 et n: `1 + rand() / (RAND_MAX + 1.0) * n`.
- Factoriser au maximum le code, en particulier celui commun aux différentes opérations logiques en utilisant des fonctions représentant l'opération a effectuer sur les élements des matrices opérandes et ceci sans utiliser de structures de contrôle.

1. Implémenter cette classe.
2. Ecrire un programme prenant en argument les tailles N_1, M_1, N_2, M_2 de deux matrices ainsi qu'un modulo n qui teste toutes les fonctionalités de cette classse et qui effectueles opérations sur une matrice $N_1 \times M_1$ et $N_2 \times M_2$ de manière à produire un résultat semblable à:

```
The modulus is 5

one
1 3 1 1
3 2 4 2
1 0 1 0

two
1 4 2 3 2
0 1 0 4 2
0 0 2 0 2

one + two
2 2 3 4 2
3 3 4 1 2
1 0 3 0 2

one - two
0 4 4 3 3
3 1 4 3 3
1 0 4 0 3

one x two
1 2 2 3 0
0 2 0 3 0
0 0 2 0 0
```