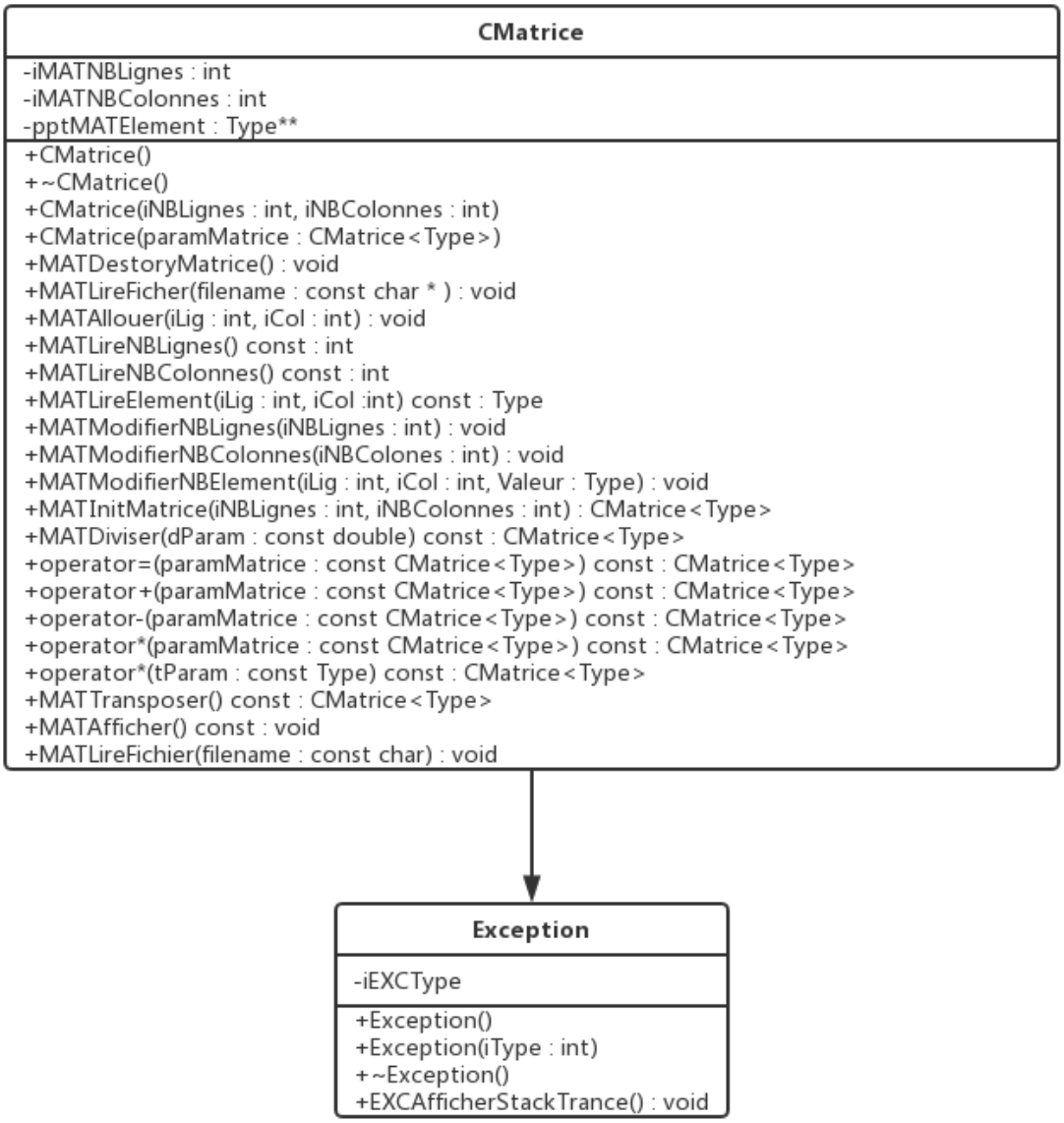
Rapport de Matrice

Élèves : Yaqi JIANG, Huajuan LI – DI 3 Mundus

Tuteur : M. Lei SHANG

1)Diagramme de classe



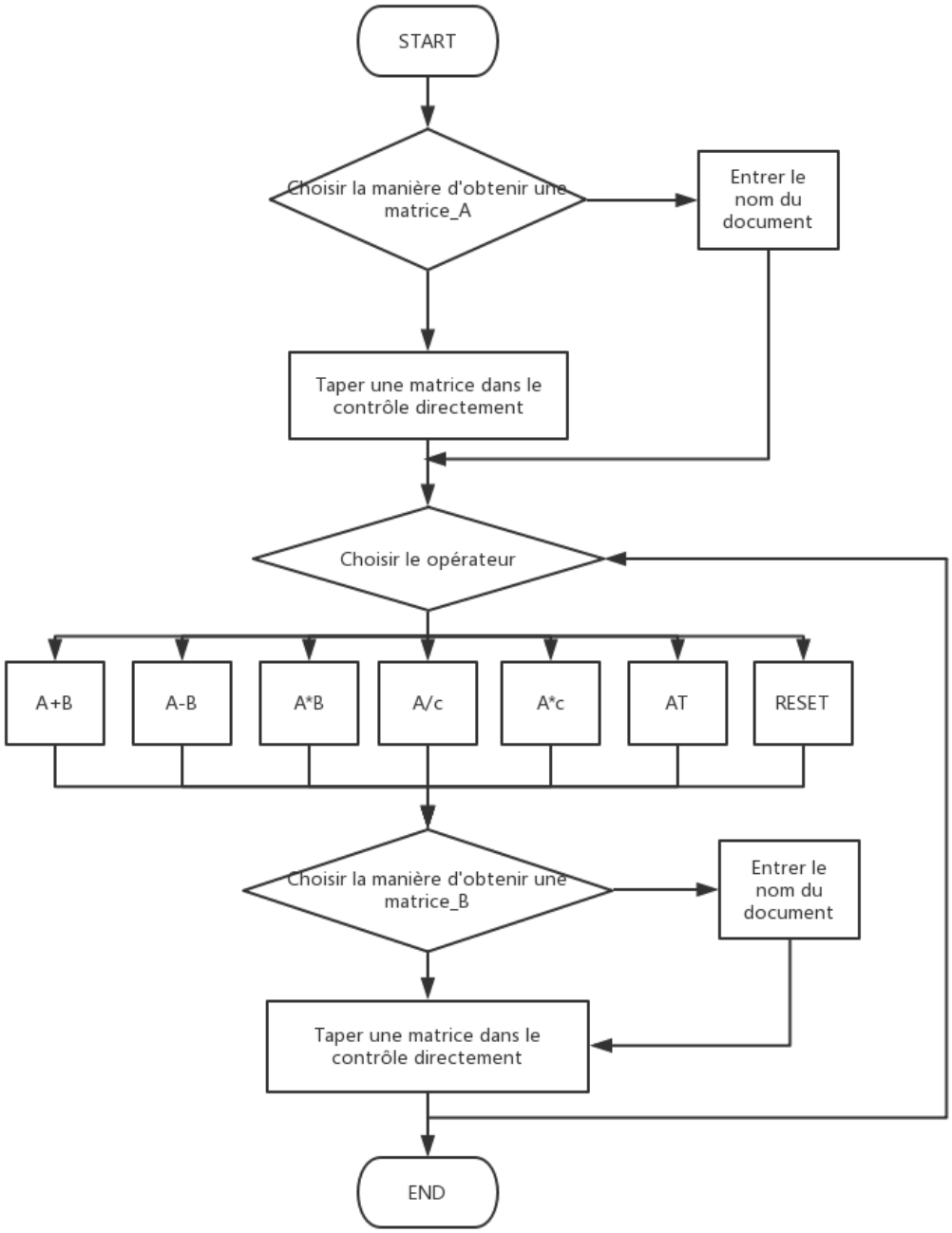
Noté **:** Ce template de classe est le cœur de la librairie. Il est composé de trois attributs : deux int pour stocker le nombre de lignes et colonnes de la matrice, et un tableau 2D stockant les éléments de la matrice.

A part la méthode pour transposer la matrice et la méthode pour diviser une matrice par un nombre, l'utilisateur pourra utiliser la matrice avec les opérateurs basiques du C++ :

- '+' : Addition de matrices de même type / classe
- '-' : Soustraction de matrices de même type / classe
- '*' : Multiplication de la matrice par une constante de type / classe compatible et produit de matrices de même type / classe
- '=' : Affectation d'une matrice dans une autre matrice de même type / classe Les opérations ci-dessus (transposée incluse, affectation exclues) ne modifient pas l'objet en cours.

Cmatrice utilise la classe Exception pour lever des exceptions.

2)Organigramme de programmation



3)l 'introduction des fonctions

A »

```
template <typename Type>
void CMatrice<Type>::MATAllouer(int iLig, int iCol)
```

Paramètre : iLig, la ligne de la matrice
Paramètre :iCol, la colonne de la matrice
Signification : C'est pour allouer la mémoire d'une nouvelle matrice, il est similaire aux constructeur avec d'arguments. Parce qu'il y a plusieurs des fonctions qui ont besoin d'allouer la mémoire, on crée cette fonction pour la faire utilisée plusieurs fois. Donc on pourrait éviter le code répétitif.

B »

```
template <typename Type>
void CMatrice<Type>::MATDetruireMatrice()
```

Signification : Il est pareil comme le destructeur, on l'utilise aussi chez le destructeur directement. Mais le destructeur est un peu limite. Quand on a besoin de détruire le mémoire avant « new », il faut réécrire des codes pour détruire au début, par exemple, an début de la fonction « MATLireFicher ». Donc, on écrire ça, et puis l'utilise aussi dans le destructeur.

C »

```
template <typename Type>
CMatrice<Type> CMatrice<Type>::MATInitMatrice(int iNBLignes, int
iNBColonnes)
```

Paramètre : iNBLignes la ligne que l'on voudrait créer
Paramètre : iNBColonnes la colonne que l'on voudrait créer
Signification : Il y a deux manières pour obtenir la matrice, on écrit ça pour initialiser une matrice vient de contrôle.

D »

```
template <typename Type>
void CMatrice<Type>::MATLireFichier(const char * filename)
```

Paramètre : filename le nom de document
Signification : Il est pour obtenir une matrice vient de document. Dans ce cas-là, on utilise « string » pour lire le contenu ligne par ligne.

4)Analyse de l'Exception

On utilise l'énumération pour marquer tous les erreurs qui apparaitraient si possible. Il est plus facile pour noter et utiliser dans notre projet. Ci-dessous est pour lister 10 types d'exception que l'on trouve.

```
a) AllouerEchoue
```

Quand on utilise « new » pour allouer la mémoire, il est possible d'échouer, par exemple, dans tous les constructeurs. Alors, on utilise ça pour rappeler l'utilisateur que vous êtes tombé en construire par « new ».

```
b) TypeMatriceDouble
```

Quand on lit le document pour obtenir une matrice, la première ligne est « TypeMatrice=double », il demande le type de « double ». Alors, si on ne trouve pas « double/Double », il montrera cette exception.

```
c) DiviseurZero
```

Quand on divise une matrice par un nombre, le nombre n'est pas 0.

```
d) ColouLigDifferent
```

En l'addition ou la soustraction, les deux matrice doit avoir la colonne égale et la ligne égale.

```
e) ColEgalePasLig
```

Quand on fait le produit de deux matrices, la colonne de matrice_1 doit égaler la ligne de matrice_2

```
f) FicherOuvertErreur
```

Quand on voudrait ouvrir le fichier pour obtenir une matrice, il échoue. Par exemple, on tombe à écrire le nom de fichier, il n'y a pas de ficher dans le directory courant, etc.

```
g) FicherVide
```

Le ficher est vide.

```
h) FicherContenuErreur
```

Il y a des erreurs dans le contenu de fichier. Les cas sont listé ci-dessous.
Cas 1 : Il n'y a pas de 4 « = » en total dans le fichier ou pas de « = » sur chaque ligne.
Cas 2 : Le contenu que l'on lit n'est pas chiffre après « NBLignes= » ou « NBColonnes= ».
Cas 3 : La quatrième ligne n'est pas « Matrice=[».
Cas 4 : La dernière ligne n'est pas «] ».

```
i) LireLigEchoue
```

Lire la ligne échoue. S'il y plus ou moins de lignes que l'orignal, il y aurait cette exception. Parce que on obtient le standard chez ligne 2-3 dans le ficher.

```
j) LireColEchoue
```

Lire la colonnes échoue. S'il y plus ou moins de colonnes que l'orignal, il y aurait cette exception. Parce que on obtient le standard chez ligne 2-3 dans le ficher.

5)Matériel de référence

a) ouvrir le fichier : http://www.cplusplus.com/reference/fstream/ifstream/is_open/
b) getline : http://www.cplusplus.com/reference/string/string/getline/
c) string : http://www.cplusplus.com/reference/string/string/
d) énumération : https://blog.csdn.net/fx677588/article/details/52679803