

Rapport : Projet C++: Les Matrices

1. Préambule

Ce projet tuteuré nous allons travailler les matrices, L'objectif est de réaliser une librairie de classes et fonctions permettant de manipuler des matrices d'éléments . La réalisation de ce projet va nécessiter l'utilisation de la méthodologie UML pour modéliser le logiciel a développer ,ainsi que le langage C++ pour la mise en oeuvre ^[1] , nous utilisons le VS 2012 pour C++.

Dans ce projet nous créons deux classes pour gérer des matrices : CMatrice et Cexception. CMatrice stocke les attributs de matrice, les méthodes des opérations et des gestions .CException stocke les méthodes des exceptions, par exemple , erreur d'ouvre le fichier ...

Nous utilisons la classe patron CMatrice pour réaliser la condition de créer des objets Matrice pour lequel le type des éléments est quelconque.

Nous créons <mystring.h> pour la solution de gérer les problèmes de string, il y a trois fonctions :

1). <bool ComparerStringAvecLong(char*str1, char*str2, int eLength)>: comparer deux strings avec certain long.

2). <int TailleDeString(char* str) >: retourner la taille de string.

3). <bool ComparerAvecString(char*str1, char*str2)>: comparer deux strings sans certain long.

2. Structure UML

Il y a deux classes dans ce projet <<Class CMatrice>> et <<Class Cexception>>. La relation entre les deux classes est la relation d'association.

2.1 Le diagramme de classes(figure 2.1)



figure-2.1

3. L'Explication de diagramme

3.1 La classe CMatrice

Le nom de cette classe est <CMatrice>. Un patron de classe peut être vu comme un modèle pour la création dynamic de classe similaire. Donc nous utilisons le mot clef <template> pour créer cette classe. Nous utilisons cette classe pour stocker les informations de matrice.

3.1.1 attributs:

Cette classe comporte quatre attributs. La visibilité est <private>. Ils sont:
int eMATLignes, int eMATColonnes, char *pcMATTypeMatrice, MType **pMATMatrice.

- 1) int eMATLignes: Cet attribut représente le nombre de lignes de matrice.
- 2) int eMATColonnes: Cet attribut représente le nombre de colonnes de matrice.
- 3) char *pcMATTypeMatrice: Cet attribut représente le type de éléments de matrice.
- 4) MType **pMATMatrice: Cet attribut représente les éléments de matrice. Pour gérer plusieurs type de matrices, cet attribut est défini par <typename MType>.

3.1.2 methodes:

Cette classe comporte quinze méthodes. La visibilité est <public>.

- 1). CMatrice <MType> ();
 - Fonction: le constructeur
- 2). ~CMatrice <MType> ();
 - Fonction: le destructeur
- 3). CMatrice <MType> (CMatrice <MType> &MATarg);
 - Fonction: Le constructeur de copie
- 4). void MATModifierType(char* pcTypeArg);
 - Les paramètres: un pointeur sur le type char
 - Fonction: modifier le attribut pcTypeArg de classe CMatrice
 - Retour: void
- 5). void MATModifierLigne(int eLigne);
 - Les paramètres: un entier
 - Fonction: modifier le attribut eMATLignes de classe CMatrice
 - Retour: void
- 6). void MATModifierColonne(int eColonne);
 - Les paramètres: un entier
 - Fonction: modifier le attribut eMATColonnes de classe CMatrice

- Retour: void
- 7). void MATModifierMatrice(MType **MTYMatrice);
 - Les paramètres: un pointeur sur le pointeur de type MType
 - Fonction: modifier le attribut pMATMatrice de classe CMatrice
 - Retour: void
 - 8). typename CMatrice<MType>& MATMultiplier(MType &MTYvalue);
 - Les paramètres: un nombre
 - Fonction: multiplier une matrice par un nombre, cette opération ne modifier pas le contenu de matrice originale
 - Retour: un objet de classe CMatrice
 - 9). typename CMatrice<MType>& MATDiviser(MType &MTYvalue);
 - Les paramètres: un nombre
 - Fonction: diviser une matrice par un nombre, cette opération ne modifier pas le contenu de matrice originale
 - Retour: un objet de classe CMatrice
 - 10). typename CMatrice<MType>& MATTransposer();
 - Fonction: claculer sa matrice transposée, cette opération ne modifier pas le contenu de matrice originale
 - Retour: un objet de classe CMatrice
 - 11). typename CMatrice<MType>& operator+(CMatrice<MType> &MATarg);
 - Les paramètres: un objet MATarg de classe CMatrice
 - Fonction: faire l'addition de deux classes, cette opération ne modifier pas le contenu de matrice originale
 - Retour: un objet de classe CMatrice
 - 12). typename CMatrice<MType>& operator-(CMatrice<MType> &MATarg);
 - Les paramètres: un objet MATarg de classe CMatrice
 - Fonction: faire la soustraction de deux classes, cette opération ne modifier pas le contenu de matrice originale
 - Retour: un objet de classe CMatrice
 - 13). typename CMatrice<MType>& operator*(CMatrice<MType> &MATarg);
 - Les paramètres: un objet MATarg de classe CMatrice
 - Fonction: faire le produit de deux classes, cette opération ne modifier pas le contenu de matrice originale
 - Retour: un objet de classe CMatrice
 - 14). void MATAffichier();
 - Fonction: afficher à l'écran une matrice
 - Retour: void
 - 15). CMatrice<double>& MATLire(char *pcFileNom);

- Les paramètres: le chemin d'un fichier
- Fonction: lire le contenu d'une matrice dans un fichier texte et créer un objet de classe CMatrice en mémoire pour stocker les données lues dans le fichier
- Retour: un objet de classe CMatrice associe

3.2 La classe CException

Le nom de cette classe est <CException>. Nous utilisons cette classe pour gérer les exception.

3.2.1 attributs:

Cette classe comporte un attribut. La visibilité est <private>. Ils sont: int eErreur.

1). int eErreur: cet attribut comporte le numérotation d'exceptions. Chacun représente les différents types d'erreur.

3.2.2 méthodes

Cette classe comporte huit méthodes. La visibilité est <public>.

- 1). Cexception();
 - Fonction: le constructeur
- 2). Cexception(int arg);
 - Fonction: le constructeur de copie
- 3). int &CEXLireErreur();
 - Fonction: Lire l'attribut eErreur d'objet
 - Retour: Une référence de type int
- 4). void CEXOuvreFile();
 - Fonction: Si le fichier ne peut pas ouvrir, il va throw une exception et afficher le raison d'exception.
 - Retour: void
- 5). void CEXType();
 - Fonction: si le type de matrice qui est dans le fichier n'est pas double, il va throw une exception et afficher le raison d'exception.
 - Retour: void
- 6). void CEXFileType();
 - Fonction: si le contenu de fichier n'est pas correct, il va throw une exception et afficher le raison d'exception. Par exemple, les informations de matrice ne sont pas assez .
 - Retour: void

7). void CEXCalculer();

- Fonction: si il y a un problème de mathématiques, il va throw une exception et afficher le raison d'exception. Par exemple, diviser une matrice par zéro.

- Retour: void

8). void CEXErreurGeneral();

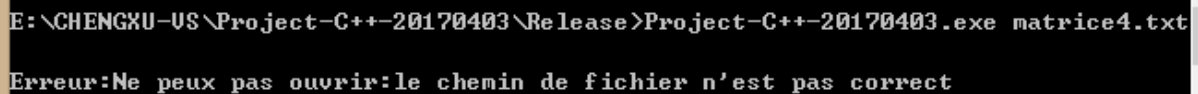
- Fonction: si il y a un problème de mémoire, il va throw une exception et afficher le raison d'exception.

- Retour: void

4. Résultat

4.1 Les exceptions

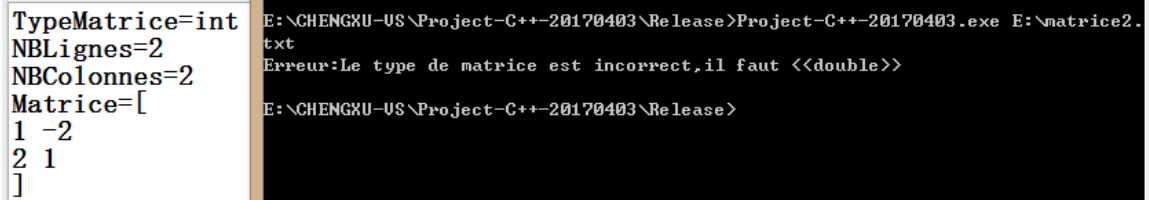
Si le chemins des fichiers sont faux ,il y a une exception.(figure4.1.1)



```
E:\CHENGXU-US\Project-C++-20170403\Release>Project-C++-20170403.exe matrice4.txt
Erreur:Ne peux pas ouvrir:le chemin de fichier n'est pas correct
```

figure4.1.1


Si le type de matrice est incorrect, il y a une exception.(figure4.1.2)



```
TypeMatrice=int
NBLignes=2
NBColones=2
Matrice=[
1 -2
2 1
]
E:\CHENGXU-US\Project-C++-20170403\Release>Project-C++-20170403.exe E:\matrice2.txt
Erreur:Le type de matrice est incorrect,il faut <<double>>
```

figure4.1.2

Si le contenu de fichier est incorrect, il y a une exception.(figure4.1.3)



```
TypeMatrice=double
NBLignes=3
NBColones=3
Matrice=[
6.34 8 1098.3243
36.454 -212.345 534.645
]
E:\CHENGXU-US\Project-C++-20170403\Release>Project-C++-20170403.exe E:\matrice3.txt
Erreur:Le contenu de fichier est incorrect,verifiez votre fichier,SUP
```

figure4.1.3

Si il est en contradiction avec règles mathématiques, il y a une exception.(figure4.1.4)

```

Ecrire une constante pour multiplier et diviser:
0
Multiplier:le resultat de Matrice est:
0      0      0      0      0
0      0      0      0      0
0      -0     0      0      0
0      0      0      0      0
0      -0     0      0      0
0      0      0      0      0
0      -0     0      0      0
0      0      0      0      0
0      0      0      0      0
0      0      0      0      0
0      -0     0      0      0
0      0      0      0      0
0      -0     0      0      0
0      0      0      0      0
0      -0     0      0      0
0      0      0      0      0
0      -0     0      0      0
0      0      0      0      0
-----
Diviser:le resultat de Matrice est:
Erreur:Il est en contradiction avec les regles de math

```

figure4.1.4

4.2 Test

Nous entrons le fichier de <release>et le fichier <.exe> par la commande ,et puis nous le passons de paramètres plusieurs fichiers par les chemins des fichiers.

Après la presse de <enter>, il affiche les matrices originales et les matrices transposées automatiquement. (figure4.2.1)

```
E:\CHENGKU-US>Project-C++-20170403\Release>Project-C++-20170403.exe E:\matrice3.txt
txt E:\matrice5.txt E:\matrice6.txt
La matrice originale est:
6.34      8      10.32
36.4      -22.3     5.5
-12      10.16     20.09

Transposer:le resultat de Matrice est:
6.34      36.4      -12
8      -22.3     10.16
10.32     5.5      20.09

La matrice originale est:
7.04      -12.19     1.05
-3.3      6.6      -12.16
20.17     -3.23     3.24

Transposer:le resultat de Matrice est:
7.04      -3.3      20.17
-12.19     6.6      -3.23
1.05      -12.16     3.24

La matrice originale est:
-6.34      8      -10.32
36.4      -22.3     5.5
-12      10.16     -20.09

Transposer:le resultat de Matrice est:
-6.34      36.4      -12
8      -22.3     10.16
-10.32     5.5      -20.09

TypeMatrice=double
NBLignes=3
NBColonnes=3
Matrice=[
7.04 -12.19 1.05
-3.3 6.6 -12.16
20.17 -3.23 3.24
]
```

figure4.2.1

Ensuite, il présente < Ecrire une constante pour multiplier et diviser : > dans l'écran pour importer une constante <c> à faire des opérations <élémentaires>. Si c=0, il y a une exception pour l'opération <diviser>. Après la presse de <enter>, il montre les résultats des opérations. (figure4.2.2)

```

=====
Ecrire une constante pour multiplier et diviser:
2.5
Multiplier:le resultat de Matrice est:
15.85      20      25.8
91         -55.75   13.75
-30        25.4    50.225
=====
Diviser:le resultat de Matrice est:
2.536      3.2      4.128
14.56      -8.92    2.2
-4.8       4.064    8.036
=====
Multiplier:le resultat de Matrice est:
17.6       -30.475   2.625
-8.25      16.5     -30.4
50.425     -8.075   8.1
=====
Diviser:le resultat de Matrice est:
2.816      -4.876   0.42
-1.32      2.64     -4.864
8.068      -1.292   1.296
=====
Multiplier:le resultat de Matrice est:
-15.85     20      -25.8
91         -55.75   13.75
-30        25.4    -50.225
=====
Diviser:le resultat de Matrice est:
-2.536     3.2      -4.128
14.56      -8.92    2.2
-4.8       4.064    -8.036
=====

```

figure4.2.2

Et puis il aussi affiche le résultat de l'addition de toutes les matrices entrées, le résultat de l'opération : $M_1 - M_2 + M_3 \dots$ et le résultat du produit des matrices. Si il est en contradiction avec les règles de math, il y a une exception.

Après il présent <continuer ou quitter> pour refaire le processus ou quitter. (figure4.2.3)

```

=====
L'addition et soustraction:le resultat de Matrice est:
-7.04      28.19    -1.05
76.1       -51.2    23.16
-44.17     23.55    -3.24
=====
Le produit:le resultat de Matrice est:
-8787.5611  6286.9118   -5273.1277
7797.3252   -6317.4494  6146.001
-12340.257  8656.6519   -5202.3771
=====
continuer ou quitter?
quitter

```

figure4.2.3

5. Compréhension

Quand nous faisons le projet, nous rencontrons des problèmes avec la mémoire . Notre programme est normale dans la mode debug, mais nous ne pouvons pas l'ouvre dans la mode release.

Enfin ,nous pensons que il y a erreur à désallouer de la mémoire quand nous faisons <new>. C'est un bon projet pour apprendre des connaissances de C++ et gérer les pointeurs . C'est très important que nous comprenons la mécanisme de pointeur pour C et C++.

6. Référence

[1] ^{MR} Vincent T'kindt, Le document de «Projet C++ : Les matrices» ,Ecole Polytechnique de l'Université de Tours , 03/2017.