TP – ExamManager
ING1 – Programmation C
Génie Informatique

Modalités

— Durée : 2 heures
— Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
— Aucun document papier n'est autorisée.
— Aucun déplacement n'est autorisé.
— Aucune question au professeur n'est autorisé.
— Aucun échange, de quelle que nature que ce soit, n'est possible.
— La présence d'un Makefile et des commentaires doxygens sont un plus, et seront appréciés.
— La lisibilité du code et les commentaires seront pris en compte dans la notation, de même que la séparation des différentes procédures/fonctions dans des fichiers d'en-tête (.c .h).
— Tout code qui ne compile pas entrainera une **pénalité de 50%. Si une partie de votre code ne compile pas, commentez-le.** Ceci implique évidemment que vous compiliez au fur et à mesure!
— Chaque warning de compilation entrainera une **pénalité de 15%**.
— Tout ce qui sera dans le dossier **rendu** correspondra à votre rendu, et sera récupéré à la fin de l'examen. Je vous conseille vivement de travailler directement dans ce dossier.

Inversion de matrice

Réaliser un programme qui permet d'inverser une matrice $\mathbf{A}$, avec $\mathbf{A}$ une matrice dans $\mathbb{R}^n \times \mathbb{R}^n$, si l'inverse existe (dans le cas contraire, un message sera affiché).

Le programme pourra calculer l'inverse selon deux méthodes : la méthode du *déterminant*, et la méthode de *Gauss*. Le choix de la méthode se fera lors de l'exécution du programme, au niveau des arguments. La dimension de la matrice à inverser sera également donné en argument du programme. Le premier argument sera la dimension de la matrice [1], et le 2e argument sera le nom de

1. Pour rappel, la méthode `strtol` permet de convertir une chaîne de caractères en entier.

la méthode.

Par exemple, pour exécuter le programme avec la méthode de Gauss pour une matrice de dimension 4, la ligne de commande pourra être :

```
./progInversion 4 gauss
```

Le programme demandera la saisie de la matrice **A**. Elle pourra être automatisée par la redirection de flux. Par exemple, le programme pourra être appelé de la façon suivante :

```
cat matriceA-3.txt | ./progInversion 3 determinant
```

avec le fichier "matriceA-3.txt" ayant le contenu suivant :

```
3 1 2
1 1 1
4 1 3
```

Généralités sur les matrices

(1) Implémenter la fonction **float** allouerMatrice(int int_n, int int_m)** qui permet d'allouer l'espace mémoire pour une matrice de dimension (int_n, int_m). □

(2) Implémenter la procédure **void initMatrice(float** ppflt_matrice, int int_n)** qui permet d'initialiser chaque valeur de la matrice carrée en fonction de ce que l'utilisateur aura saisi. □

(3) Implémenter la procédure **void afficherMatrice(float** ppflt_matrice, int int_n)** qui permet d'afficher la matrice carrée. □

(4) Implémenter la procédure **void libererMatrice(float** ppflt_matrice, int int_n)** qui permet de libérer l'espace mémoire de la matrice. □

Méthode du déterminant

Une première méthode pour calculer l'inverse d'une matrice **A** inversible se fait par le déterminant $det(\mathbf{A})$ et la comatrice $com(\mathbf{A})$ avec la formule suivante :

$$\mathbf{A}^{-1} = \frac{1}{det(\mathbf{A})} {}^t com(\mathbf{A})$$

Le calcul du déterminant peut se faire de façon récursive, suivant une ligne i (ou une colonne) :

$$det(\mathbf{A}) = \sum_{j=1}^{n} a_{ij}(-1)^{i+j} det(\mathbf{A}_{ij})$$

et chaque coefficient $c_{i,j}$ de la comatrice $com(A)$ se calcule comme suit :

$$c_{i,j} = (-1)^{i+j} det(\mathbf{A}_{ij})$$

avec $\mathbf{A}_{ij}$ la matrice **A** privée de la ligne i et la colonne j.

(5) Implémenter la fonction **`float** extraireMatrice(float** ppflt_matrice, int int_n, int i, int j)`** qui renvoie une nouvelle matrice correspondant à la matrice `ppflt_matrice` privée de la ligne i et la colonne j. □

(6) Implémenter la fonction **`float determinant(float** ppflt_matrice, int int_n)`** qui calcule le déterminant de la matrice. □

(7) Implémenter la fonction **`float** comatrice(float** ppflt_matrice, int int_n)`** qui renvoie la comatrice de `ppflt_matrice`. □

(8) Implémenter la fonction **`float** inverseDeterminant(float** ppflt_matrice, int int_n)`** qui renvoie l'inverse de `ppflt_matrice`. □

Méthode de Gauss

Une autre méthode pour calculer l'inverse d'une matrice est l'algorithme du pivot de Gauss. Le principe consiste à appliquer sur une matrice identité (de même dimension que la matrice **A**) la même suite d'opérations élémentaires qui permettra de transformer la matrice **A** en matrice identité. Une fois la matrice **A** transformée en matrice identité, la matrice identité qui a subi les

mêmes opérations sera l'inverse de **A**. Le découpage proposé est le même que celui vu en TD d'Algorithmique.

(9) Implémenter la procédure **void dilatation(float** ppflt_matrice, int int_n, int int_m, int i, float k)** qui prend en paramètre une matrice carrée (ou rectangulaire horizontale) et qui la modifie en effectuant l'opération élémentaire $L_i \leftarrow kL_i$. □

(10) Implémenter la procédure **void permutation(float** ppflt_matrice, int int_n, int int_m, int i, int j)** qui prend en paramètre une matrice carrée (ou rectangulaire horizontale) et qui la modifie en effectuant l'opération élémentaire $L_i \leftrightarrow L_j$. □

(11) Implémenter la procédure **void transvection(float** ppflt_matrice, int int_n, int int_m, int i, int j, float k)** qui prend en paramètre une matrice carrée (ou rectangulaire horizontale) et qui la modifie en effectuant l'opération élémentaire $L_i \leftarrow L_i + kL_j$. □

(12) Implémenter la procédure **void zeroSousPivot(float** ppflt_matrice, int int_n, int int_m, int i)** qui place des 0 sous le "pivot". L'élément `ppflt_matrice[i][i]` est le "pivot". Il est supposé non nul. Par une dilatation appropriée, on le transforme en 1. Puis, par des transvections successives, on place des 0 sous le pivot. □

(13) Implémenter la procédure **void formeTriangulaire(float** ppflt_matrice, int int_n, int int_m)** qui transforme la matrice en matrice triangulaire supérieure. Pour cela, on itère la procédure `zeroSousPivot`. Si le pivot est nul, permuter la ligne du pivot avec la première des lignes suivantes qui ne comporte pas de 0 à la place considérée. Si toute la colonne en dessous du pivot nul est composée de 0, la matrice n'est pas inversible. □

(14) Implémenter la procédure **void zeroSurPivot(float** ppflt_matrice, int int_n, int int_m, int i)** qui place des 0 sur le "pivot". La matrice `ppflt_matrice` est sensée avoir été obtenue par la procédure `formeTriangulaire`. Elle est donc triangulaire supérieure avec des 1 sur la diagonale. Par des transvections successives, on place des 0 sur le "pivot" `ppflt_matrice[i][i]`. □

15 Implémenter la procédure **void identite(float** ppflt_matrice, int int_n, int int_m)** qui transforme la matrice en matrice identité. Pour cela, on itère la procédure `zeroSurPivot`. □

16 Implémenter la fonction **float** inverseGauss(float** ppflt_matrice, int int_n)** qui renvoie l'inverse de `ppflt_matrice`. Pour cela :

1. on concatène la matrice `ppflt_matrice` et la matrice identité ;
2. on effectue la méthode `formeTriangulaire` afin d'obtenir une matrice triangulaire supérieure avec des 1 sur la diagonale ;
3. on effectue la méthode `identite` de manière à obtenir la matrice identité ;
4. enfin, on retourne la moitié droite de la matrice obtenue, qui contient l'inverse de la matrice `ppflt_matrice` initiale. □

Exemple de déroulé :

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix}$$

Contaténation de A et de I : $\begin{bmatrix} 1 & 1 & 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 & 1 & 0 \\ 2 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$

$$\begin{matrix} L_2 \leftarrow L_2 - L_1 \\ L_3 \leftarrow L_3 - 2L_1 \end{matrix} \quad \begin{bmatrix} 1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 1 & 0 \\ 0 & -1 & -3 & -2 & 0 & 1 \end{bmatrix}$$

$$L_3 \leftarrow L_3 + L_2 \quad \begin{bmatrix} 1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 1 & 0 \\ 0 & 0 & -4 & -3 & 1 & 1 \end{bmatrix}$$

$$L_3 \leftarrow \tfrac{-1}{4} L_3 \quad \begin{bmatrix} 1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 1 & 0 \\ 0 & 0 & 1 & 3/4 & -1/4 & -1/4 \end{bmatrix}$$

$$\begin{matrix} L_1 \leftarrow L_1 - 2L_3 \\ L_2 \leftarrow L_2 + L_3 \end{matrix} \quad \begin{bmatrix} 1 & 1 & 0 & -1/2 & 1/2 & 1/2 \\ 0 & 1 & 0 & -1/4 & 3/4 & -1/4 \\ 0 & 0 & 1 & 3/4 & -1/4 & -1/4 \end{bmatrix}$$

$$L_1 \leftarrow L_1 - L_2 \quad \begin{bmatrix} 1 & 0 & 0 & -1/4 & -1/4 & 3/4 \\ 0 & 1 & 0 & -1/4 & 3/4 & -1/4 \\ 0 & 0 & 1 & 3/4 & -1/4 & -1/4 \end{bmatrix}$$

$$\mathbf{A^{-1}} = \begin{bmatrix} -1/4 & -1/4 & 3/4 \\ -1/4 & 3/4 & -1/4 \\ 3/4 & -1/4 & -1/4 \end{bmatrix}$$