Travaux Pratiques de langage C n°5

Cours d'Informatique de l'École Centrale d'Électronique

—Ingénieur Première Année—

Nous allons voir comment représenter des matrices sous forme de tableaux à deux dimensions, afin d'avoir à notre disposition un petit "*package*" de manipulation de matrices (utile par exemple pour effectuer des transformations dans l'espace à trois dimensions). Les sujets abordés dans ce TP sont les suivants :

- comment saisir une matrice sur des types simples ;
- comment effectuer les opérations de base sur les composantes d'une matrice ;
- comment effectuer des opérations algébriques sur les matrices.

Dans ce qui suit, on considère que les matrices manipulées sont codées par des tableaux à deux dimensions `TAILLE_MAX` × `TAILLE_MAX`, où `TAILLE_MAX` sera une valeur entière de votre choix. Il est fortement conseillé d'écrire une fonction d'affichage de matrice.

On aura aussi tout intérêt à donner un code de retour pour chaque fonction qui ne renvoie rien, afin de savoir si elle a échoué ou non. On appelle **vecteur** un tableau à 1 dimension de taille maximale `TAILLE_MAX`.

▸ **Exercice 1. (saisie d'une matrice d'entiers)** *Écrire la fonction* `LireMat` *qui prend en paramètre un tableau à deux dimensions, deux entiers, et qui remplit le tableau avec des entiers lus au clavier de la façon suivante :*

- *les deux entiers n et m pris en paramètre devront contenir respectivement le nombre de lignes et le nombre de colonnes de la matrice à la fin de la saisie ;*
- *la saisie s'arrête lorsque $n \times m$ valeurs entières ont été saisies.*

▸ **Exercice 2. (recopies d'objets)** *Lorsqu'on modifie un tableau on peut vouloir faire des opérations sur une recopie de ce tableau. Pour cela, nous avons besoin de deux fonctions :*

- `RecopVect`, *qui prend deux vecteurs et la taille du premier en argument et qui recopie le premier dans le second ;*
- `RecopMat`, *qui prend deux matrices en argument, ainsi que les dimensions de la première et qui recopie la première dans la seconde.*

À partir de maintenant, on ne vous spécifie plus les précautions à prendre en matière de recopie d'objets ni d'arguments. Les vérifications de cohérence sont aussi à faire.

▸ **Exercice 3. (opérations élémentaires)** *Voici quelques opérations élémentaires sur les matrices :*

- *écrire les fonctions* `AddVect`, `SousVect` *qui additionne et soustrait (resp.) deux vecteurs composante à composante ; écrire aussi la fonction* `MultVect` *qui multiplie les composantes d'un vecteur par un entier ;*
- *écrire la fonction* `EchangeLigne` *qui échange deux lignes d'une matrice.* `EchangeLigne` *prend en paramètre une matrice, ses dimensions et les indices des deux lignes à échanger ;*
- *écrire les fonctions* `AddMat`, `SousMat` *qui additionne et soustrait (resp.) deux matrices composante à composante ; écrire aussi les fonctions* `MultMatEnt`, `MultMatVect` *et* `MultMatMat` *qui multiplient (à gauche) une matrice par respectivement un entier, un vecteur et une matrice ;*
- *écrire la fonction* `Transpose` *qui calcule la transposée d'une matrice.*

▸ **Exercice 4. (calcul d'un déterminant)** *Détailler la méthode habituelle pour le calcul du déterminant. Écrire la fonction* `Det` *qui réalise cette méthode.*

▸ **Exercice 5. (calcul de l'inverse)** *Le calcul de l'inverse d'une matrice peut s'effectuer par l'intermédiaire des cofacteurs : l'inverse d'une matrice est exactement la transposée de la matrice des cofacteurs divisée par la valeur du déterminant.*

- *écrire la fonction* `CoFact` *qui prend en argument deux indices (entiers)* `i`, `j`, *une matrice avec ses dimensions et qui calcule le cofacteur* $c_{i,j}$ *;*
- *écrire la fonction* `CofactMat` *qui construit la matrice des cofacteurs ;*
- *donner le programme* C *de calcul de l'inverse d'une matrice.*