

Magistère de Mathématiques, 2^e année

—

TD 2 : Manipulation de matrices

Luc Bougé, Hervé Jégou

27 septembre 2004

I. Introduction

Comme lors du premier TD, l'éditeur `emacs` est recommandé et la compilation de votre code source sera effectuée en utilisant l'option `-Wall` du compilateur `gcc` :

```
gcc -Wall -o prog prog.c
```

La compilation de votre programme ne doit pas générer de warning.

Dans ce TD, vous allez manipuler des tableaux. Afin de pouvoir changer rapidement la taille de ceux-ci, vous utiliserez pour la taille des tableaux manipulés un entier N préalablement déclaré au moyen de la directive `#define` :

```
#define N 5
```

Pour visualiser les tableaux que vous allez manipuler, vous utiliserez les fonctions d'affichage fournies en annexe de ce sujet :

- La fonction `matrix_int_disp` prend en paramètre un tableau d'entier de taille N et l'affiche ;
- La fonction `matrix_double_disp_2D` prend en paramètre un tableau bi-dimensionnel de réels de taille $N \times N$ et l'affiche.

Afin de gagner du temps, vous pouvez récupérer ces fonctions à partir de l'URL suivante :

<http://www.irisa.fr/temics/Equipe/Jegou/Teaching>

Pour cela, vous pouvez utiliser dans votre butineur la fonction *Save As* du menu *File*.

II. Variables locales et globales

Considérez le programme ci-dessous.

```
#include <stdio.h>

int n = 1;
int m;

int f(int n)
{
    m = n;
    n = 3;
    printf("A: n = %d\n", n);

    {
        int n;
        n = 7;
        printf("B: n = %d\n", n);
    }
    printf("C: n = %d\n", n);
    n = 2;
    return m * 2;
}

int main(void)
{
    int p, q;

    printf("D: m = %d\n", m);
    printf("E: n = %d\n", n);
    printf("F: p = %d\n", p);

    p = 5;
    q = f(p);
    printf("G: m = %d\n", m);
    printf("H: n = %d\n", n);
    printf("I: p = %d\n", p);
    printf("J: q = %d\n", q);
    return 0;
}
```

Note : Inutile de retapez le code, il est disponible sur la page web du TD indiqué en introduction. Néanmoins, jouez le jeu! N'exécutez ce programme qu'à la fin de cette partie, dans un but de vérification.

Question II.1. *Prévoyez l'affichage généré à l'exécution.*

Question II.2. *Donnez la portée de chacune des variables déclarées.*

III. Tableau 1D et instructions de contrôle

Question III.1. *Déclarez un tableau de taille N et initialisez-le avec les N premiers entiers naturels positifs. Vérifiez que cette initialisation est correcte en utilisant la fonction d'affichage qui vous est fournie.*

Question III.2. *Soustrayez 4 à chacun des éléments du tableau précédent. Vous utiliserez ce tableau modifié pour tester les fonctions des questions suivantes.*

Question III.3. *Écrivez une fonction qui affiche les éléments d'un tableau de taille N et qui s'arrête si un élément nul est rencontré. Vous utiliserez pour cela l'instruction `break`.*

Question III.4. *En utilisant l'instruction `continue`, écrivez une fonction qui n'affiche que les éléments pairs d'un tableau d'entiers.*

IV. Tableau 2D : initialisation

Question IV.1. *Écrivez une fonction `matrix_comp` qui associe une valeur réelle à deux paramètres entiers. Dans un premier temps, la fonction sera définie telle que $\text{matrix_comp}(i,j) = \frac{2i+3j}{10.23}$.*

La fonction `matrix_comp` sera utilisée pour initialiser les valeurs de la matrice en fonction, respectivement, des indices de ligne et de colonne.

Question IV.2. *Déclarez en tant que variable globale une matrice carrée de taille $N \times N$.*

Question IV.3. *Écrivez une fonction `void matrix_init(void)` qui initialise la matrice précédemment définie avec le terme générique `matrix_comp(i,j)`.*

Question IV.4. *Écrivez une fonction `matrix_sum` qui renvoie la somme des éléments de la matrice globale.*

Note : le parcours de la matrice doit respecter l'ordre de l'allocation mémoire.

V. Matrice transposée et multiplication matricielle

Question V.1. *Écrivez une fonction `matrix_transpose` qui transpose la matrice définie globalement dans la partie précédente. Pour cela, utilisez une matrice auxiliaire `mat_aux`, définie localement dans la fonction `matrix_transpose`. Affichez le résultat avec la fonction fournie.*

Question V.2. *Bonus : même question que la précédente mais sans utiliser de matrice auxiliaire. Est-ce possible si la matrice n'est pas carrée ?*

Question V.3. *Écrivez une fonction `matrix_mul` qui prend deux matrices carrées en paramètres et qui effectue leur multiplication. Le résultat sera stocké dans la matrice globale et affiché pour vérification.*

VI. Annexe : Fonctions d'affichage de tableaux

```
/* Fonction qui prend un tableau d'entier en paramètre et qui l'affiche */
void matrix_int_disp(int tableau[N])
{
    int i;

    for (i = 0; i < N; i++)
printf("%7d ", tableau[i]);
    printf("\n");
}

/* Fonction qui prend en paramètre un tableau de réels
à deux dimensions (égales à N) et qui affiche son contenu */
void matrix_double_disp_2D(double tableau2D[N][N])
{
    int i, j;

    for (i = 0; i < N; i++) {
for (j = 0; j < N; j++)
printf("%7.3f %p", tableau2D[i][j]);

/* On retourne à la ligne pour chaque ligne du tableau */
printf("\n");
    }
}
```