

S3:E5 – Matrices et fichiers

LICENCE INFORMATIQUE : PROGRAMMATION NÉCESSAIRE

Stéfane

1 Objectif

On évite au tant que faire ce peut de modéliser les types abstraits bidimensionnels avec de la mémoire en 2D¹. On utilisera de préférence la mémoire 1D – sous forme de **vecteurs** donc – pour modéliser les TA 2D.

Par exemple une matrice $A_{M \times N}$ sera représentée par un vecteur de taille $M * N$ où les valeurs seront celles de A rangées *ligne par ligne* (ou colonne par colonne).

2 Les types abstraits et autres entêtes

Deux types abstraits vont être définis pour y ranger des *couples* de coordonnées (ligne, colonne) et pour y ranger des *matrices*.

2.1 Le TA **pair**

Créez un fichier **pair.h** contenant la déclaration d'un couple de coordonnées matricielles :

```
struct pair {  
    int l, c;  
}
```

et les déclarations suivantes :

- a) `struct pair * consPair (int l, int c);`
- b) `struct pair * cpyPair (struct pair * P);`
- c) `void freePair (struct pair * P);`

1. En l'état actuel des modèles de machines existants, la mémoire 2D n'existe pas.

d) `int pair2ind (struct pair * p, struct matrix * M);`

qui renvoie l'indice dans le vecteur `values` correspondant au couple de coordonnées référencé par `p`

e) `struct pair * ind2pair (int k, struct matrix * M);`

qui renvoie une paire `(l,c)` correspondant à l'indice `k`.

2.2 Le TA `matrix`

Créez un fichier `matrix.h` contenant la déclaration du type matrice

```
struct matrix {  
    double * values;  
    int n, m;  
}
```

et les déclarations suivantes :

a) `struct matrix * consMatrix (int n, int m);`

Alloue la mémoire et initialise les champs `m` et `n`.

b) `struct matrix * cpyMatrix (struct matrix * M);`

Créer une copie exacte de la matrice `M`.

c) `void freeMatrix (struct matrix ** M);`

Libère *toute* la mémoire occupée par `*M` et mais celle-ci à `NULL`.

d) `void viewMatrix (struct matrix * M, char * entete);`

Visualise la matrice `M` après avoir affiché son `entête`.

e) `struct matrix * matrixInput (char * filename);` qui va utiliser :

— La structures `FILE *` (cf. fichier `stdio.h` et les exemples sur Arche)

— Les opérateurs des fichiers textes (de la librairie `stdio.h`)

```
FILE *fopen(const char *pathname, const char *mode);
int fclose(FILE *stream);
int fprintf(FILE *stream, const char *format, ...);
int fscanf(FILE *stream, const char *format, ...);
int feof(FILE *stream);
int fseek(FILE *stream, long offset, int whence);
long ftell(FILE *stream);
void rewind(FILE *stream);
int fgetpos(FILE *stream, fpos_t *pos);
int fsetpos(FILE *stream, const fpos_t *pos);
```

Pour connaître leurs fonctionnements, utiliser la commande `man` ;

Le fichier d'une matrice aura la forme suivante

```
3 4
1.1 1.2 1.3 1.4
2.1 2.2 2.3 2.4
3.1 3.2 3.3 3.4
```

où la première ligne est la dimension de la matrice et les lignes suivantes les valeurs rangées... ligne par ligne

f) `struct matrix * matrixAdd (struct matrix * A, struct matrix * B);`

dont la définition est :

$$(A)_{n \times m} + (B)_{n \times m} = (a_{i,j}) + (b_{i,j}) = (c_{i,j}) = (C)_{n \times m}$$

g) `struct matrix * matrixMult (struct matrix * A, struct matrix * B);`

dont la définition est :

$$(A)_{n \times m} \cdot (B)_{m \times p} = \left(\sum_{k=1}^m a_{i,k} \cdot b_{k,j} \right) = (c_{i,j}) = (C)_{n \times p}$$

3 Les définitions des fonctions

Créer le fichier `pair.c` pour y ranger les définitions des fonctions déclarées dans `pair.h`.

Créer le fichier `matrix.c` pour y ranger les définitions des fonctions déclarées dans `matrix.h`.

4 La fonction principale

Créez un fichier `main.c` qui contiendra uniquement la fonction `main` qui devra :

- a) Appeler la fonction `matrixInput ("data/A.txt")`;
- b) Appeler la fonction `matrixInput ("data/B.txt")`;
- c) Visualiser ces deux matrices
- d) En fonction de la compatibilité des dimensions des deux matrices proposez :
l'addition, la multiplication ou bien les deux;
- e) Appelez la fonction souhaitée par l'utilisateur et visualiser le résultat.

5 Travail personnel

Complétez le code ci-dessus en permettant de sauvegarder une matrice dans un fichier au format binaire ainsi que de lire un fichier binaire contenant une matrice. Modifiez le code de la fonction principale en conséquence pour les tests.