UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence Informatique
Info0101 – 2017/2018

Travaux dirigés n° 6

Tableaux à deux dimensions

Exercice 1 (Les matrices)
Dans cet exercice, les matrices sont représentées par des tableaux à 2 dimensions.

1. Écrivez une fonction/procédure prenant en paramètre une largeur et une hauteur et créant une matrice de ces dimensions.
2. Écrivez une fonction/procédure prenant en paramètre n et qui crée la matrice identité I_n.
3. Écrivez une fonction/procédure prenant en paramètre une matrice m, deux indices i et j, et inversant les deux lignes i et j de m. Quelle modifications sont à apporter pour inverser deux colonnes ?
4. Écrivez une fonction/procédure prenant en paramètre deux matrices et retournant la somme de ces deux matrices. Quelles conditions sont à poser sur les matrices ?
5. Même chose avec la multiplication de deux matrices passées en paramètre.

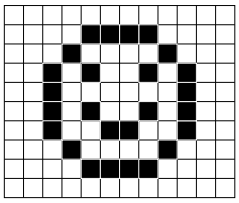
Exercice 2 (Représentation mémoire)
1°) Donnez la représentation mémoire complète aux 4 points précisés en commentaires dans l'algorithme suivant :

```
Algorithme tableau2D
Déclarations
        Variables
                t : tableaux de tableaux d'entiers
                u : tableau d'entiers
                i : entier
Début
{1}     t ← allouer(4)
{2}     u ← allouer(4) /* Point 1 */
{3}     Pour i allant de 0 à taille(u)-1 Faire
{3.1}           u[i] ← i + 1
        FinPour/* Point 2 */
{4}     Pour i allant de 0 à taille(t)-1 Faire
{4.1}           t[i] ← u
        FinPour/* Point 3 */
{5}     t[2][2] ← 5 /* Point 4 */
Fin
```

2°) Idem pour l'algorithme suivant :

```
Algorithme tableau2D
Déclarations
        Variables
                t : tableaux de tableaux d'entiers
                u : tableau d'entiers
                i, j : entier
Début
{1}     t ← allouer(4) /* Point 1 */
{2}     Pour i allant de 0 à taille(t)-1 Faire
{2.1}           t[i] ← allouer(i + 1)
        FinPour/* Point 2 */
{3}     Pour i allant de 0 à taille(t)-1 Faire
{3.1}           Pour j allant de 0 à taille(t[i])-1 Faire
{3.1.1}                 t[i][j] ← i * 10 + j
                FinPour
        FinPour/* Point 3 */
{4}     u ← t[2]
{5}     u[2] ← -1 /* Point 4 */
Fin
```

Exercice 3 (Image en noir et blanc)
Nous souhaitons représenter une image en noir et blanc à l'aide d'un tableau à 2 dimensions de booléens. Chaque case correspond à un pixel qui peut être allumé (valeur `vrai`) ou éteint (valeur `faux`). La figure suivante montre un exemple d'image *(a)*, ainsi qu'un exemple d'affichage dans l'invite de commandes *(b)* :



(a)

```
____XXXX____
___X____X___
__X_X__X_X__
__X_____X___
__X_____X___
__X_X__X_X__
__X__XX__X__
___X____X___
____XXXX____
____________
```

(b)

1°) Écrivez une fonction/procédure `effacerImage` qui prend en paramètre une image et qui efface tous les pixels.

2°) Écrivez une fonction/procédure `creerImage` qui prend en paramètre une largeur et une hauteur et qui crée une image vide (sans pixel allumé).

3°) Écrivez une fonction/procédure `afficherImage` qui prend en paramètre une image et qu'il l'affiche à l'écran : un pixel éteint est représenté par un espace et un pixel allumé par un "X".

4°) Écrivez une fonction/procédure `setPixel` qui prend en paramètre les coordonnées d'un pixel et qui l'allume sur l'image passée en paramètre. Vous devez traîter les cas où les valeurs des paramètres sont incorrectes.

5°) Écrivez une fonction/procédure `ligne` qui prend en paramètre l'indice d'une ligne et qui allume tous les pixels de la ligne concernée sur l'image passée en paramètre. Vous devez traîter les cas où les valeurs des paramètres sont incorrectes.

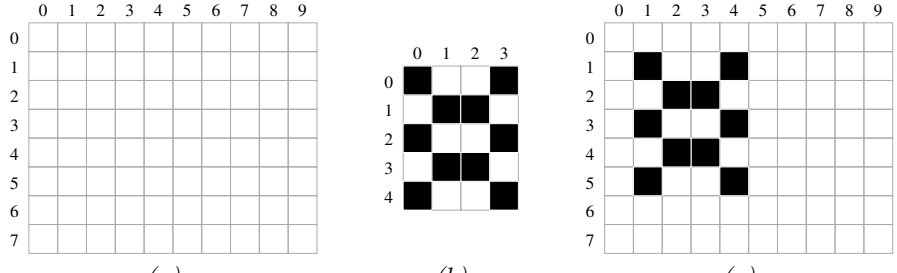
6°) Nous souhaitons écrire une fonction `cadre` qui retourne les coordonnées du rectangle encadrant les pixels allumés de l'image. Par exemple, sur l'image précédente, le rectangle possède les coordonnées (2,1)-(9,8) (ce sont les coordonnées du point en haut à gauche et du point en bas à droite). En pratique, les coordonnées d'un rectangle sont représentées par un tableau à 2 dimensions, chaque ligne correspondant aux coordonnées d'un point.

- Écrivez la fonction `premiereLigne` qui retourne l'indice de la première ligne contenant un pixel allumé (avec l'exemple précédent, elle retourne 1). Si l'image est vide, la fonction retourne 0.
- Sans donner l'algorithme, indiquez quelles sont les modifications à apporter pour écrire la fonction `premiereColonne` qui retourne l'indice de la première colonne contenant un pixel allumé.
- Écrivez la fonction `derniereLigne` qui retourne l'indice de la dernière ligne contenant un pixel allumé (avec l'exemple précédent, elle retourne 8). Si l'image est vide, la fonction retourne l'indice de la dernière ligne.
- En supposant la fonction `dernierеColonne` donnée, écrivez la fonction `cadre`.
- Écrivez le code *Java* de la fonction `cadre`.

7°) Nous souhaitons écrire une fonction/procédure `inverser` qui prend en paramètres une image et les coordonnées d'un rectangle et qui inverse, dans l'image, tous les pixels qui font partie du rectangle. Vous gèrerez les cas d'erreur.

8°) Écrivez un algorithme principal qui permet de tester **toutes** les fonctions/procédures réalisées précédemment.

9°) Écrivez une fonction/procédure qui prend en paramètre deux images, ainsi que les coordonnées d'un point et qui recopie la deuxième image dans la première à la position donnée. L'exemple ci-dessous montre l'image *(b)* qui est copiée dans l'image *(a)* à la position (1,1), ce qui donne l'image *(c)*.



(a) *(b)* *(c)*