SÉANCE 4

Objectif

Le but de cette quatrième séance est de poursuivre la manipulation des tableaux et des entrées-sorties simples.

Outils

On souhaite manipuler des images de niveaux de gris en utilisant des tableaux à deux dimensions. On choisit de représenter une image de niveaux de gris par un tableau dont les éléments (les pixels) sont de type `unsigned char`. Un niveau de gris est donc un entier appartenant à l'intervalle $[0, 255]$ où 0 représente le noir et 255 le blanc.

Il existe de nombreux formats de fichiers contenant des images. Nous allons ici considérer le format « PGM-ASCII ». Il s'agit de fichiers de texte contenant :

- les deux caractères `P2` (c'est la « signature » du format PGM-ASCII) ;
- un ou plusieurs séparateurs (espace, tabulations ou sauts de lignes) ;
- le nombre de colonnes de l'image ;
- un ou plusieurs séparateurs ;
- le nombre de lignes de l'image ;
- un ou plusieurs séparateurs ;
- la valeur du plus grand niveau de gris de l'image ;
- un ou plusieurs séparateurs ;
- les niveaux de gris de l'image séparés par un ou plusieurs séparateurs, disposés du coin en haut à gauche de l'image au coin en bas à droite, ligne par ligne (dans l'ordre de lecture d'un texte français).

Comme vous n'avez pas encore vu la manipulation des fichiers, vous allez écrire votre programme en considérant que le contenu du fichier est tout simplement tapé au clavier par l'utilisateur.

La fonction `scanf` utilisée avec le format `%s` pour lire une chaîne de caractères s'arrête dès qu'elle rencontre un séparateur. Utilisée avec les formats `%d` pour lire un `int` (pour le nombre de colonne et le nombre de lignes de l'image) ou `%hhu` pour lire un `unsigned char` (pour le plus grand niveau de gris et le niveau de gris de chaque pixel), la fonction `scanf` ignore (« saute ») les éventuels séparateurs qui précèdent le nombre à lire.

Exercice

Écrivez un programme qui :

1. lit le contenu d'un fichier au format PGM-ASCII tapé au clavier par l'utilisateur et stocke les niveaux de gris des pixels dans un tableau à deux dimensions ;
2. dans un second tableau à deux dimensions, écrit les niveaux de gris des pixels de l'image après avoir subi une rotation de 90° (voir l'exemple ci-dessous) ;
3. affiche à l'écran le contenu du fichier au format PGM-ASCII correspondant à l'image ayant subi la rotation.

Si on effectue une rotation de 90° au tableau suivant :

1	2	3
4	5	6

on obtient le tableau suivant :

3	6
2	5
1	4

Sous Linux, afin d'éviter de devoir taper toutes les informations au clavier à chaque exécution du programme, vous pouvez utiliser la redirection de l'entrée standard (`stdin`), associée au clavier, vers un fichier.

Vous pouvez tester votre programme sur l'image contenue dans le fichier `feep.pgm`. Cette image contient 7 lignes et 24 colonnes. Pour cela, après l'avoir récupérée depuis Moodle et placée dans le même répertoire que celui de votre programme, si votre exécutable s'appelle `rotation`, vous pouvez lancer l'exécution de votre programme en tapant :

```
./rotation < feep.pgm
```

Vous devriez obtenir le résultat suivant :

```
P2
7 24
15
  0   0   0   0   0   0   0
  0  15  15  15   0   0   0
  0  15   0  15   0   0   0
  0  15   0  15   0   0   0
  0  15  15  15  15  15   0
  0   0   0   0   0   0   0
  0   0   0   0   0   0   0
  0  11   0   0   0  11   0
  0  11   0  11   0  11   0
  0  11   0  11   0  11   0
  0  11  11  11  11  11   0
  0   0   0   0   0   0   0
  0   0   0   0   0   0   0
  0   7   0   0   0   7   0
  0   7   0   7   0   7   0
  0   7   0   7   0   7   0
  0   7   7   7   7   7   0
  0   0   0   0   0   0   0
  0   0   0   0   0   0   0
  0   3   0   0   0   0   0
  0   3   0   3   0   0   0
  0   3   0   3   0   0   0
  0   3   3   3   3   3   0
  0   0   0   0   0   0   0
```

Pour voir le contenu du fichier image initial, vous pouvez taper :

```
cat feep.pgm
```

Pour visualiser le résultat sous forme d'image, vous pouvez rediriger la sortie standard (`stdout`), associée à l'écran, vers un autre fichier. Le lancement de l'exécution du programme devient alors :
`./rotation < feep.pgm > feep2.pgm`
Si vous disposez de l'outil ImageMagick, alors il suffit de taper :
`display feep2.pgm`
qui affichera dans une petite fenêtre l'image obtenue.

Vous pouvez tester votre programme sur des images plus grandes en utilisant par exemple le fichier `chien.pgm` qui contient une image de 300 lignes et 400 colonnes.

Pour aller plus loin

Vous pouvez également utiliser l'image de votre choix. Pour cela, vous devez la convertir en image de niveaux de gris au format PGM-ASCII. Avec ImageMagick, pour convertir par exemple le fichier `mon_fichier_image.jpg` en `mon_fichier_image.pgm`, vous pouvez utiliser la commande suivante :

`convert mon_fichier_image.jpg -compress none mon_fichier_image.pgm`
Mais, attention, il est très probable qu'apparaissent des commentaires dans l'en-tête du fichier qui contient l'image. Il s'agit de lignes qui commencent par le caractère `#`. Vous devrez alors modifier votre programme pour qu'il ignore ces lignes.