TD/TP 9 - Lecture, traitement et écriture d'images au format **PGM** et **PPM**

Le but de ce projet est de développer des programmes de traitement d'images pouvant lire, traiter et écrire des fichiers images au format **PGM** binaire et **PPM** binaire (auteur : Jef Poskanzer, http://www.acme.com/).

Le format **PNM** (Portable aNyMap) regroupe trois formats de fichiers images, à savoir le Portable BitMap file format (**PBM**), le Portable GrayMap file format (**PGM**) et le Portable PixMap file format (**PPM**). Chacun de ces trois formats de fichiers a une variante binaire ou ASCII. L'entête des fichiers **PBM**, **PGM** ou **PPM**, toujours écrit en ASCII est structuré de la même manière :

- 1 nombre magique du format sur 2 octets : de P1 à P6
- Des commentaires éventuels : #
- 2 entiers indiquant la hauteur et la largeur de l'image (en nombre de pixels)
- 1 entier indiquant la valeur maximale (profondeur) des pixels

Concernant le nombre magique :

- **PBM** (Portable BitMap) - image binaire : P1 en ASCII, P4 en binaire,
- **PGM** (Portable GrayMap) - image en niveaux de gris : P2 en ASCII, P5 en binaire,
- **PPM** (Portable PixMap) - image en couleur (vraies couleurs, 3 plans R, G, B) : P3 en ASCII, P6 en binaire.

Concernant la valeur maximale des pixels (celle-ci n'existe pas pour le format **PBM**), chaque pixel est codé par 1 ou 2 octets selon que la valeur maximale soit strictement inférieure (1 octet), ou bien supérieure ou égale à 256 (2 octets). Avec le format **PGM**, c'est en général la valeur 255 qui est utilisé afin de coder un pixel sur 1 octet soit 256 niveaux de gris. Pour le format **PPM**, chaque valeur R, G, B, est codée sur 1 ou 2 octets selon que la valeur maximale soit strictement inférieure ou bien supérieure ou égale à 256.
Dans ce projet nous travaillerons spécifiquement sur les fichiers au format **PGM** binaire (P5) et **PPM** binaire (P6) pour des images composés d'octets non-signés (valeurs de 0 à 255). Le fichier **PGM** binaire est constitué d'une entête ASCII suivi des octets de l'image parcourue ligne par ligne de la gauche vers la droite. L'entête a le format suivant (toujours en ASCII) :

```
P5
# commentaires
300 700
255
```

Ce projet est constitué, en plus des fichiers images, du fichier `image.h` (fournit sous Moodle), du fichier `image.c` (à écrire) et de programmes de tests permettant de répondre à chacune des questions.

Exercice 1 Ouverture, lecture et écriture d'une image au format PGM

L'objectif de cet exercice est d'écrire un programme principal `test_pgm.c` permettant de lire une image au format **PGM**, copier l'image lue dans une seconde image et de l'écrire (`$./test_pgm 01_Lena.pgm 01_Lena_out.pgm`).
Dans le fichier `image.h` un nouveau type `typedef unsigned char OCTET;` est déclaré et est utilisé pour déclarer les tableaux dynamiques d'images : `OCTET *ImgIn, *ImgOut;`.

Afin de lire une image au format **PGM** il faut dans un premier temps lire l'entête du fichier image afin de connaître la taille (hauteur × largeur) de l'image, effectuer une allocation dynamique d'un tableau d'`OCTET` et lire les valeurs des pixels de l'image.

Dans le fichier `image.c` écrire la fonction `void lire_nb_lignes_colonnes_image_pgm(char nom_image[], int *nb_lignes, int *nb_colonnes);` permettant d'allouer dynamiquement le tableau de l'image lue `OCTET *ImgIn`. Une fois l'allocation dynamique effectuée, écrire la fonction `void lire_image_pgm(char nom_image[], OCTET *pt_image, int taille_image);`.

Une fois l'allocation dynamique du tableau de l'image écrite `OCTET *ImgOut`, écrire la fonction `void ecrire_image_pgm(char nom_image[], OCTET *pt_image, int nb_lignes, int nb_colonnes);`. En fin du programme principal, ne pas oublier de libérer les mémoires.

Tester votre programme avec l'image `01_Lena.pgm` (à ouvrir au préalable avec un éditeur d'images et un éditeur de texte afin de visualiser l'entête).
Tester votre programme avec l'image `02_Girafes.pgm`. Écrire une fonction `void ignorer_commentaires(FILE * f);` qui sera ajoutée dans les 3 fonctions précédemment écrites pour ignorer les commentaires éventuels entre le nombre magique et la taille de l'image.

Exercice 2 Traitement d'images PGM

L'objectif de cet exercice est d'écrire 3 fonctions qui pour chacune d'entre elle, traite une image lue en entrée (avec en argument d'entrée d'éventuels paramètres nécessaires pour le traitement) et enregistre le résultat obtenu dans une image de sortie.

Écrire un programme `seuil_pgm.c` qui à partir d'une image **PGM** en entrée et d'une valeur d'un seuil (compris entre 0 et 255) génère une image seuillée **PGM** pour laquelle tous les pixels ayant une valeur inférieure au seuil sont mis à 0, ou 255 sinon. La valeur du seuil sera un argument d'entrée du programme.

Écrire un programme `inverse_pgm.c` qui lit une image **PGM** en entrée, et gènère une image **PGM** dans laquelle les niveaux de gris sont inversés.

Écrire un programme `contraste_pgm.c` qui à partir d'une image **PGM** en entrée et d'une valeur alpha (comprise entre -255 et 255) génère une image **PGM** plus claire ou plus foncée en fonction de la valeur d'alpha qui sera un argument d'entrée du programme.

Exercice 3 Ouverture et écriture d'une image au format PPM

Le format **PPM** est un **PGM** "couleur" dans lequel chaque pixel est codé par trois octets consécutifs représentant les composantes R, G, B. L'objectif de cet exercice est d'écrire les fonctions permettant de lire et écrire une image couleur au format **PPM**.

Écrire dans le fichier `image.c` les fonctions `void lire_nb_lignes_colonnes_image_ppm(char nom_image[], int *nb_lignes, int *nb_colonnes);`, `void lire_image_ppm(char nom_image[], OCTET *pt_image, int taille_image);` et `void ecrire_image_ppm(char nom_image[], OCTET *pt_image, int nb_lignes, int nb_colonnes);`. L'allocation dynamique d'une image couleur est 3 fois plus grande que celle d'une image en niveaux de gris.

Tester ces fonctions dans programme principal `test_ppm.c` permettant de lire une image au format **PPM**, ce copier son contenu dans une seconde image et de l'écrire. tester ce programme avec l'image `03_kodim15_girl.ppm`.

Exercice 4 Seuillage des 3 composantes d'une image couleur `PPM`

Écrire un programme `seuil_ppm.c` qui à partir d'une image **PPM** en entrée et de 3 valeurs de seuil (compris entre 0 et 255) génère une image seuillée **PPM** pour laquelle toutes les composantes R ayant une valeur inférieure au seuil rouge sont mis à 0, ou 255 sinon (idem pour les composantes G et B). Les trois valeurs de seuil seront des arguments d'entrée du programme.
Tester ce programme avec l'image `03_kodim15_girl.ppm`.

Exercice 5 Conversion d'une image couleur `PPM` en une image en niveaux de gris `PGM`

L'objectif de cet exercice est de convertir une image couleur (R, G, B) au format `PPM` en une image en niveaux de gris (Y) au format `PGM` suivant la transformation suivante :

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B. \quad (1)$$

Tester ce programme avec l'image `04_Lena.ppm`. Comparer le résultat obtenu avec l'image `01_Lena.pgm`.