Mini-projet **ITII1**
Algorithmique **2018-2019**

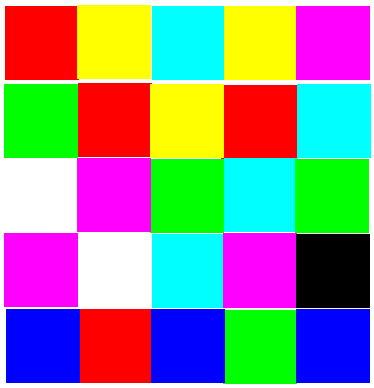
Partie 1 : Prise en main des images

On dispose d'une image BMP couleur codée sur 24 bits. Le but de ce mini-projet est de réaliser un outil de traitement d'images pour des fichiers BMP de dimension 2Nx2N.

1. En utilisant les fonctions de BMPLib, récupérez le contenu d'une image BMP dans une structure donneesImageRGB contenant le nombre de lignes hauteurImage et de colonnes largeurImage de l'image et un tableau d'octets. Ce tableau contient une suite d'octets, qui peuvent être regroupés trois par trois : BVR. Le premier triplet correspond au premier point de la première ligne de l'image. Le second triplet au deuxième point de la première ligne, etc.
2. Sur un écran d'ordinateur, les couleurs sont codées sur trois composantes (rouge-vert-bleu). Chaque composante est elle-même codée sur un octet non signé (unsigned char) et prend des valeurs comprises entre 0 et 255. Cela permet d'avoir un choix de 16 millions de couleurs (256^3). Le tableau suivant donne quelques exemples de couleurs obtenues en combinant les trois composantes RVB :

Rouge	255	0	255	255	0	255	0	0	128
Vert	0	255	255	0	0	255	255	0	128
Bleu	0	0	0	255	0	255	255	255	128
Couleur	rouge	vert	jaune	magenta	noir	blanc	cyan	bleu	gris

Le codage des couleurs pour les images suit le même principe. Ainsi, dans un fichier image BMP couleur codé sur 24 bits, chaque point de l'image est représenté par un triplet (r, v, b). Le fichier BMP contient donc une suite d'octets. Les trois premiers octets de la suite correspondent au premier point de la première ligne, les trois suivants au deuxième point de la première ligne, et ainsi de suite. Les points de la deuxième ligne sont stockés à la suite du dernier point de la première ligne, et ainsi de suite. L'ordre de stockage des composantes est inversé, c'est-à-dire que les composantes d'un point sont stockées dans l'ordre bvr, et non rvb.



Cette image correspond à la suite d'octets :

0 0 255, 0 255 255, 255 255 0, 0 255 255, 255 0 255, 0 255 0, 0 0 255, 0 255 255, 0 0 255, 255 255 0, 255 255 255, 255 0 255, 0 255 0, 255 255 0, 0 255 0, 255 0 255, 255 255 255, 255 255 0, 255 0 255, 0 0 0, 255 0 0, 0 0 255, 255 0 0, 0 255 0, 255 0 0

Les virgules n'existent pas dans le fichier. Elles ont été ajoutées dans l'exemple pour faciliter la lecture.

Afin de lire et écrire des fichiers BMP couleur sur 24 bits, on dispose des fonctions du fichier bmplib.h suivant :

```
/* Veuillez reporter tout commentaire a ghislain.oudinet@isen.fr */

#ifndef BMPLIB_H
#define BMPLIB_H

#include <stdbool.h>

/* Partie publique elaboree
   Cette partie est relative a toutes les informations utiles pour
   personnaliser et mieux controler l'interaction avec les fichiers BMP */

typedef struct
{
      int largeurImage;
      int hauteurImage;
      unsigned char *donneesRGB; /* Donnees sauvees dans l'ordre bleu-vert-rouge */
} DonneesImageRGB;

/* Fonction s'occupant de la liberation complete d'une structure DonneesImageRGB */
void libereDonneesImageRGB(DonneesImageRGB **structure);

/* Fonction essayant de lire le fichier passe en parametre, et renvoyant une structure
      contenant les informations de l'image en cas de succes, NULL sinon */
DonneesImageRGB *lisBMPRGB(char *nom);

/* Fonction ecrivant les informations de l'image dans le fichier passe en parametre.
      Renvoie faux en cas de probleme, vrai sinon */
bool ecrisBMPRGB_Dans(DonneesImageRGB *donneesImage, char *nom);

#endif
```

Ce fichier header décrit donc les fonctions permettant de gérer des images BMP.

1. Tester les fonctions de lecture et d'écriture d'une image BMP à l'aide d'une image. On appellera d'abord la fonction de lecture, puis celle d'écriture en changeant le nom de l'image. Sur le disque, l'image doit être dupliquée.

2. On veut définir trois matrices de 2^Nx2^N valeurs entières, représentant les trois composantes rouge, vert et bleu de l'image. Ecrire la fonction *cree3matrices* qui alloue dynamiquement les trois matrices, puis les remplit à partir des valeurs de l'image DonneesImageRGB *image. Pour chaque case, la valeur stockée dans la matrice rouge est la composante rouge du point correspondant dans l'image. De même pour les matrices verte et bleue.
   Dans l'exemple précédent, les trois matrices sont de taille 5x5 et contiennent les valeurs suivantes :

   Rouge

255	255	255	255	255
0	255	255	255	0
255	255	0	0	0
255	255	0	255	0
0	255	0	0	0

   Vert

0	255	255	255	0
255	0	255	0	255
255	0	255	255	255
0	255	255	0	0
0	0	0	255	0

   Bleu

0	0	255	0	255
0	0	0	0	255
255	255	0	255	0
255	255	255	255	0
255	0	255	0	255

   Ecrire la fonction *creeMatrices* qui transforme les données du champ donneesRGB d'une structure `DonneesImageRGB *image` en trois matrices de short int rouge, vert et bleu
3. Ecrire la fonction *creeImage* qui transforme trois matrices de 2^Nx2^N valeurs entières en `DonneesImageRGB *image`. C'est l'opération duale de l'opération de la question 2. Attention aux allocations dynamiques de mémoire…
4. Modifier la fonction *main* pour qu'elle appelle les deux fonctions précédentes et tester le programme.

Partie 2 : bases du traitement d'images

Dans cette partie, plusieurs outils de base du traitement des images seront mis en œuvre. Ces outils travaillent uniquement sur des matrices et non directement sur `DonneesImageRGB *image`

1. Négatif

Afin de créer le négatif d'une image, on inverse chacune des composantes, c'est-à-dire que :

$$r \rightarrow 255\text{-}r$$
$$v \rightarrow 255\text{-}v$$
$$b \rightarrow 255\text{-}b$$

Ecrire la fonction *negatifImage* qui crée les trois matrices correspondant à l'image en négatif, à partir des matrices d'une image donnée

2. Transformation en niveaux de gris

Afin de transformer une image couleur (trois matrices) en une image en niveaux de gris (une matrice), on utilise la relation suivante :

$$g = 0.2125\ r + 0.7154\ v + 0.0721\ b$$

Ecrire la fonction *couleur2NG* qui crée la matrice en niveaux de gris correspondant aux trois matrices RVB d'une image couleur.

3. Seuillage

Pour cette question, on considère une image en niveaux de gris. Seuiller une image en niveaux de gris consiste à la binariser, c'est-à-dire qu'une valeur de seuil s est choisie entre 0 et 255, puis toutes les valeurs supérieures à s deviennent égales à 255, tandis que les valeurs inférieures prennent la valeur 0.

Ecrire la fonction *seuillageNG* qui crée la matrice correspondant à l'image seuillée.

4. Gestion du menu

Ecrire la fonction *choixMenu* qui affiche le menu suivant à l'écran :

```
Traitement d'images
1 - négatif d'une image
2 - transformation en niveaux de gris d'une image couleur
3 - seuillage
0 - quitter
```

et saisit un entier compris entre 0 et 3. Cet entier est le choix de l'utilisateur.

Ecrire la fonction *traiteChoix* qui permet de gérer chaque valeur de choix : saisie des données, appel de la fonction correspondante et sauvegarde du résultat.