SÉANCE 10

Objectif

Le but de cette séance est de poursuivre la mise en œuvre des techniques de programmation modulaire en modifiant le module `image` fait lors de la séance 8 et utilisé lors de la séance 9. Mais, si vous n'avez pas terminé ces deux séances, faites-le avant de poursuivre.

Exercice

Exercice (Lecture d'une image dans un format binaire)

L'objectif de cet exercice est de prendre en compte, en plus du format PGM-ASCII, le format PGM-BINAIRE.

Le format PGM-BINAIRE est identique au format PGM-ASCII sauf :

- la « signature » du format PGM-BINAIRE qui est constituée des deux caractères `P5` au lieu de `P2` pour le format PGM-ASCII ;
- les niveaux de gris de l'image qui sont représentés en binaire, chaque niveau de gris étant codé sur un octet [1], et qui sont, comme pour le format PGM-ASCII, 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).

1. Modifier la fonction `ImLire` pour qu'elle puisse, en plus d'une image au format PGM-ASCII, lire aussi une image au format PGM-BINAIRE. Dans les deux cas, l'en-tête du fichier étant du texte, le fichier doit être ouvert en mode texte. Mais, si l'image est au format PGM-BINAIRE, alors, pour lire les niveaux de gris, la position courante dans le fichier doit être mémorisée, le fichier doit être fermé, il doit être rouvert au format binaire et vous devez retourner à la position mémorisée afin de pouvoir faire appel à la fonction `fread`. Effectuez la lecture des niveaux de gris ligne par ligne, un appel à `fread` permettant de lire une ligne. On rappelle que si `Pix` est une variable de type `Matrice`, alors `Pix[i]` contient l'adresse du début de la ligne numéro `i` de la matrice.
2. Pour tester la nouvelle version de la fonction :
   - copiez le fichier `tp9ex.c` de la séance 9 dans le fichier `tp10ex.c` ainsi que le fichier de description des dépendances associé ;
   - utilisez le fichier `chenille.pgm` qui est au format PGM-BINAIRE ;
   - au lieu de mettre les noms des fichiers directement dans `tp10ex.c`, transmettez-les sous la forme de paramètres passés au programme.
3. Ajouter au module une fonction d'en-tête :
   ```
   void ImEcrireBin(Image Im, char NomFichier[])
   ```
   qui écrit l'image `Im` dans le fichier de nom `NomFichier` au format PGM-BINAIRE. Pour cela, après avoir écrit l'en-tête dans le fichier ouvert en mode texte, fermez le fichier et rouvrez-le en mode binaire avec écriture en fin de fichier pour écrire les niveaux de gris, ligne par ligne, avec la fonction `fwrite`. Modifiez le fichier `tp10ex.c` pour tester cette fonction.

1. On suppose que le niveau de gris maximal est inférieur ou égal à 255 (il existe une variante de ce format où chaque niveau de gris est représenté par deux octets).

Pour aller plus loin

Si vous avez terminé l'exercice précédent, pour aller plus loin, après avoir fait une sauvegarde de vos fichiers, vous pouvez ajouter la prise en compte des images en couleur. Une des manières de représenter la couleur de chaque pixel avec trois composantes consiste à donner son niveau de rouge, son niveau de vert et son niveau de bleu (espace RVB ou RGB), chacun de ces niveaux étant un entier compris entre 0 et 255.

Il existe de nombreux formats de fichiers contenant des images en couleur. Le format « PPM-ASCII » est l'extension à la couleur du format « PGM-ASCII ». Il s'agit de fichiers de texte contenant les mêmes informations, avec les exceptions suivantes :

- la signature du format PPM-ASCII est constituée des deux caractères `P3` ;
- à la place du plus grand niveau de gris de l'image, se trouve la valeur du plus grand niveau parmi les trois composantes de tous les pixels ;
- à la place du niveau de gris de chaque pixel se trouvent le niveau de rouge, suivi du niveau de vert, suivi du niveau de bleu, séparés par un ou plusieurs caractères séparateurs.

Par exemple, le fichier `imagette.ppm` :

```
P3
4 5
255
  0   0   0    0   0   0    0   0   0  255   0 255
  0   0   0    0 255 127    0   0   0    0   0   0
  0   0   0    0   0   0    0 255 127    0   0   0
255   0 255    0   0   0    0   0   0    0   0   0
255   0   0    0 255   0    0   0 255  255 255 255
```

contient une image couleur composée de 5 lignes et 4 colonnes dont le pixel situé à la fin de la première ligne a un niveau de rouge égal à 255, un niveau de vert égal à 0 et un niveau de bleu égal à 255 (couleur magenta). Le fichier `chenille.ppm` est un autre exemple de fichier au format PPM-ASCII .

Il existe également la version binaire de ce format, PPM-BINAIRE dont la signature est `P6` et où chaque pixel est représenté par trois[2] octets (niveaux de rouge, de vert et de bleu). Le fichier `chenille-bin.ppm` est un exemple de fichier au format PPM-BINAIRE.

Vous pouvez faire une nouvelle version des modules `image` et `ti` afin de prendre en compte la couleur. Pour cela, vous devez :

- modifier la structure de données du module `image` :
  - dans la partie publique (fichier d'en-tête), ajoutez :

```
enum eType // Types possibles pour une image
{
  Gris,    // Image de niveaux de gris
  Couleur // Image en couleur
};
typedef enum eType tType;
```

  - dans la partie privée (fichier d'implémentation), utilisez :

```
struct sImage
{
  int NbLig; // Nombre de lignes de l'image
  int NbCol; // Nombre de colonnes de l'image
  tType Type; // Type de l'image : Gris ou Couleur
```

2. Il existe des fichiers PPM-BINAIRE où chaque niveau est codé sur deux octets. Mais vous ne prendrez pas en compte cette possibilité.

```
    Matrice Rouge; // Matrice des niveaux de rouge ou des niveaux de gris
    Matrice Vert;  // Matrice des niveaux de vert
    Matrice Bleu;  // Matrice des niveaux de bleu
};
```

Pour une image de niveaux de gris, l'idée est ici d'utiliser la matrice des niveaux de rouge pour stocker les niveaux de gris et de ne pas allouer de place mémoire pour les deux autres matrices.

- ajouter des fonctions spécifiques aux images en couleur :
  - `ImAllouerCoul` qui crée une image couleur ;
  - `ImRouge` qui retourne la matrice des niveaux de rouge de l'image ;
  - `ImVert` qui retourne la matrice des niveaux de vert de l'image ;
  - `ImBleu` qui retourne la matrice des niveaux de bleu de l'image ;
- modifier le corps de certaines fonctions afin de prendre en compte la couleur tout en permettant aux programmes existants utilisant le module `image` pour manipuler des images de niveaux de gris de continuer à fonctionner sans devoir les modifier ; il s'agit des fonctions : `ImAllouer`, `ImLiberer`, `ImNivGris`, `ImLire` et `ImEcrire`.

Vous pouvez ensuite étendre aux images en couleur les fonctions du module `ti`. Pour l'application d'une transformation ponctuelle, dans le cas d'une image couleur, vous pouvez la réaliser sur chacune des composantes de l'image. Vous pouvez également ajouter d'autres transformations en laissant libre cours à votre imagination.