INFO0030 : Projet de Programmation
Librairie de Gestion d'Images

B. Donnet, K. Edeline, E. Marechal
Université de Liège

Alerte : Évitez de perdre bêtement des points...

— Nous vous conseillons **vivement** de relire et d'appliquer les guides de style et de langage, mis à votre disposition sur eCampus (INFO0030, Sec. Supports pour le Cours Théorique). Cela vous permettra d'éviter de nombreuses erreurs et de perdre des points inutilement.
— Nous vous conseillons de consulter la grille de cotation utilisée pour la correction des projets afin d'éviter de perdre bêtement des points. Elle est disponible sur eCampus (INFO0030, Sec. Procédures d'Évaluation).
— Votre code sera compilé et testé sur la machine virtuelle qui vous a été fournie et qui sert de référence. Elle est disponible sur eCampus (INFO0030, Sec. Projets). Veillez donc à ce que votre code fonctionne dans cet environnement.

1 Contexte

Dans le cadre du cours INFO0030, Projet de Programmation, il sera demandé, à ultérieurement, de manipuler des fichiers d'image au format `PNM` (cfr. Sec. 2).

Afin de faciliter l'implémentation des futurs projets, le projet actuel a pour but d'introduire le format `PNM`. Il vous est demandé d'écrire une bibliothèque C permettant de manipuler (chargement depuis un fichier, manipulation en mémoire, sauvegarde dans un fichier) des images au format `PNM`.

2 Portable Anymap

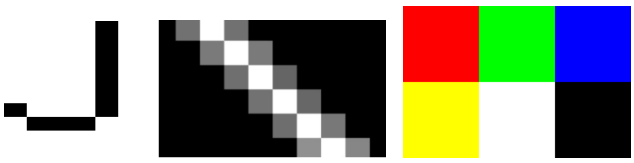
Le format d'image `PNM` (*Portable Anymap*) désigne un ensemble de formats de fichiers graphiques utilisés pour les échanges. On considère trois formats :

— `PBM` (*Portable Bitmat File Format* – Sec. 2.1).
— `PGM` (*Portable Graymap File Format* – Sec. 2.2).
— `PPM` (*Portable Pixmap File Format* – Sec. 2.3).

L'idée derrière le format `PNM` est que chaque image est décrite dans un fichier ASCII [1], facilitant ainsi la transmission électronique de l'image mais aussi toute forme de changement dans le formatage. Il est bon de noter que le format `PNM` accepte aussi l'encodage sous la forme d'un fichier binaire. Nous nous focaliserons, dans le cadre du cours INFO0030, uniquement sur le format ASCII.

Une image `PNM` peut être visualiée avec un logiciel adéquat, comme par exemple Gimp [2] ou ImageMagick. [3]

1. The American Standard for Information Interchange. Les codes ASCII représentent du texte dans, entre autre, des ordinateurs. Voir http://en.wikipedia.org/wiki/ASCII pour plus de détails.
2. The Gnu Image Manipulation Program. Voir http://www.gimp.org/.
3. Voir http://www.imagemagick.org.



(a) PBM (b) PGM (c) PPM

FIGURE 1 – L'image correspondant au fichier PNM

2.1 Portable Bitmap File

Le format PBM (*Portable Bitmap File Format*) est utilisé pour des images en noir et blanc. Un fichier PBM est un fichier composé de deux parties : un en-tête et une matrice dont chaque élément représente un pixel[4] (noir ou blanc).

L'en-tête du fichier PBM contient deux lignes :

— la première ligne est toujours la même. Il s'agit d'un nombre magique[5] identifiant le fichier PBM : P1.

— la seconde ligne contient deux entiers : le nombre de colonnes et de le nombre de lignes de pixels de l'image. Les deux nombres sont, bien entendu, séparés par un caractère d'espacement.

Les lignes qui suivent l'en-tête donnent l'information pour chaque pixel. Ceux-ci sont énumérés ligne par ligne et colonne par colonne, les différents pixels sur une ligne étant séparés par un espace. Ainsi, la première valeur est le pixel du coin supérieur gauche, la seconde est le pixel de la première ligne, deuxième colonne, etc. Le caractère '1' est utilisé pour coder un pixel noir, tandis que le pixel blanc est représenté par un caractère '0'. Toutes les lignes commençant par '#' sont ignorées.

Ci-dessous, un exemple de fichier PBM qui représente la lettre 'J'. L'image résultante (agrandie) est donnée à la Fig. 1(a).

```
P1
# Ceci est un commentaire
6 10
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
1 0 0 0 1 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

2.2 Portable Graymap File

Un fichier PGM (*Portable Gray Map Format*) est utilisé pour les images en niveau de gris. Un fichier PGM est composé de deux parties : un en-tête et une matrice dont chaque élément représente le niveau de gris d'un pixel de l'image. Dans le cadre de ce cours, nous considérons qu'il y a 256 niveaux de gris allant de 0 pour le noir à 255 pour le blanc.

L'en-tête contient trois lignes :

— La première ligne est toujours la même. Il s'agit d'un nombre magique identifiant le fichier PGM : P2.

4. "Le *pixel* (souvent abrégé px) est l'unité de base permettant de mesurer la définition d'une image numérique matricielle." Voir https://fr.wikipedia.org/wiki/Pixel

5. Un *nombre magique* peut être vu (entre autre) comme une constante déterminant un format de fichier. Voir http://fr.wikipedia.org/wiki/Nombre_magique_(programmation).

— La seconde ligne contient deux entiers : le nombre de colonnes et le nombre de lignes de pixels de l'image. Les deux nombres sont, bien entendu, séparés par un caractère d'espacement.
— Enfin, la troisième ligne contient la valeur maximale que peut prendre un pixel (255 dans notre cas)

Les lignes qui suivent donnent l'intensité de chaque pixel. Ceux-ci sont énumérés ligne par ligne et colonne par colonne, les différents pixels sur une ligne étant séparés par un espace. Ainsi, la première valeur est l'intensité du pixel du coin supérieur gauche, la seconde est l'intensité du pixel de la première ligne, deuxième colonne, etc. Toutes les lignes commençant par '#' sont ignorées.

Par exemple, un fichier `PGM` contenant les informations ci-dessous est l'encodage d'une image représentant une barre oblique blanche sur fond noir (cfr. Fig. 1(b)).

```
P2
10 6
# Ceci est un commentaire
255
0 114 255 114 0 0 0 0 0 0
0 0 123 255 123 0 0 0 0 0
0 0 0 112 255 101 0 0 0 0
0 0 0 0 115 255 103 0 0 0
0 0 0 0 0 111 255 121 0 0
0 0 0 0 0 0 145 255 135 0
```

2.3 Portable Pixmap File

Un fichier `PPM` (*Portable Pixmap File Format*) est utilisé pour les images en couleur. Un fichier `PPM` est composé de deux parties : un en-tête et une matrice dont chaque élément représente la couleur d'un pixel de l'image. La couleur d'un pixel est donnée par la combinaison de trois valeurs entières : rouge, vert et bleu.

L'en-tête contient trois ligne :
— La première ligne est toujours la même. Il s'agit d'un nombre magique identifiant le fichier `PPM` : `P3`.
— La seconde ligne contient deux entiers : le nombre de colonnes et le nombre de lignes de pixels de l'image. Les deux nombres sont, bien entendu, séparés par un caractère d'espacement.
— Enfin, la troisième ligne contient la valeur maximale utilisée pour coder les couleurs. Cette valeur doit être codée en caractère ASCII et être inférieure à 65.536.

Les lignes qui suivent donnent la couleur de chaque pixel. Ceux-ci sont énumérés ligne par ligne et colonne par colonne, les différents pixels sur une ligne étant séparés par un espace. Pour chaque pixel, il y a un triplet de valeurs (les valeurs sont séparées par un caractère d'espacement). La première valeur du triplet indique l'intensité du rouge, la deuxième valeur l'intensité du vert, et la dernière valeur l'intensité du bleu. La couleur résultante pour le pixel est donc le mélange de ces trois couleurs de base.

Le premier triplet, sur la première ligne, donne la couleur du pixel situé dans le coin supérieur gauche de l'image, le second triplet de la première ligne ligne donne le deuxième pixel (première ligne, deuxième colonne), etc. Toutes les lignes commençant par '#' sont ignorées.

Par exemple, un fichier `PPM` contenant les informations ci-dessous est l'encodage d'une image représentant plusieurs couleurs (cfr. Fig. 1(c)).

```
P3
3 2
# Ceci est un commentaire
255
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
```

3 Architecture du Code

Afin de vous aider dans votre implémentation, nous vous fournissons un squelette de programme comprenant :

— un `Makefile` permettant la compilation de votre projet ;
— les fichiers `pnm.c` et `pnm.h` permettant l'implémentation de votre bibliothèque de fichiers `PNM` ;
— le fichier `main.c` qui contient le programme principal permettant de tester votre implémentation.

Il est impératif de respecter ce squelette. Vous pouvez, si vous le jugez nécessaire, ajouter des fichiers `.h` et/ou `.c` mais, dans ce cas, vous devez penser à modifier le `Makefile` pour en tenir compte (nous ne le ferons pas à votre place !).

3.1 `Makefile`

Sur base de ce squelette, pour compiler votre projet, il vous suffit de taper la commande suivante :

```
$>make
```

Si la compilation réussit, un fichier binaire, exécutable, appelé `pnm` sera créé (voir Sec. 3.3 sur le format d'exécution du binaire `pnm`).

3.2 `pnm.{h.c}`

Le header `pnm.h` est quasiment complet. Vous devez modifier le commentaire de début afin d'y ajouter votre nom, prénom et matricule ainsi que la date de dernière modification de votre fichier.

Le header contient déjà les include guards et la déclation du type opaque `PNM` permettant de gérer une image `PNM` (`PBM`, `PGM`, ou `PPM`). Vous devrez le compléter dans le module `pnm.c`.

Les fonctions

```
int load_pnm(PNM **image, char* filename);

int write_pnm(PNM *image, char* filename);
```

sont déjà déclarées pour vous.

`load_pnm()` permet de charger en mémoire, depuis un fichier (de nom `filename`), une image `PNM`. Cette fonction devra être robuste et pouvoir faire face aux erreurs. Ainsi, `load_pnm()` renverra, en résultat, un entier. Cet entier devra suivre la nomenclature suivante :

0 . Tout s'est bien passé et l'image est correctement chargée en mémoire dans `*image`.

-1 . Il est impossible d'allouer suffisamment de mémoire pour l'image (par exemple parce que l'image est trop grosse).

-2 . Le nom du fichier en input est mal formé. Par exemple, un fichier `PBM` ne se termine pas par l'extension `.pbm`.

-3 . Le contenu du fichier en input est malformé. Par exemple, il manque des informations sur les pixels ou encore l'en-tête est mal formé.

`write_pnm()` permet de sauver une image dans un fichier (de nom `filename`). A l'instar du chargement en mémoire, cette fonction renverra un entier. L'entier résultat devra suivre la nomenclature suivante :

0 . Tout s'est bien passé et l'image a pu être correctement sauvée dans un fichier.

-1 . Le nom du fichier output passé en argument n'est pas valide. Par exemple, il contient au moins un caractère interdit. [6]

-2 . L'image n'a pas pu être sauvée dans un fichier.

6. Pour information, les caractères interdits dans un nom de fichier sont les suivants : `/ \ : * ? " < > |`.

3.3 Programme Principal

Le fichier `main.c` décrit le programme principal, qui vous permettra de tester votre implémentation.

Le fichier binaire produit par la compilation (cfr. Sec. 3.1) devra pouvoir être exécuté de la façon suivante :

```
$>./pnm -f <format> -i <image_input> -o <image_output>
```

où

- `<format>` indique le format du fichier passé en argument (i.e., `PGM`, `PBM` ou `PPM`).
- `<image_input>` est un fichier `PNM` fourni en input du programme. Attention, votre programme doit *impérativement* accepter les chemins absolus et relatif des fichiers (ex : `/home/user/x/fichier.ppm` ou bien `../../fichier.ppm`).
- `<image_output>` est le fichier où il faut copier `<image_input>`.

Il est évident que votre programme doit pouvoir gérer les cas où le format indiqué du fichier (i.e., l'option `-f <format>`) ne correspond pas aux fichiers passés en argument. Par exemple, la commande

```
$>./pnm -f PGM -i escargot.ppm -o escargot_copie.ppm
```

doit retourner un message d'erreur à l'écran. Par exemple :

```
Mauvais format passé en argument.  Le fichier escargot.ppm est du type PPM
et non PGM.
```

4 Enoncé du Projet

Il est demandé de compléter le squelette de code fourni afin de permettre la manipulation d'un fichier au format `PNM`.

Votre projet devra :

- être soumis dans une archive `tar.gz`, appelée `pnm.tar.gz`, via la Plateforme de Soumission. La décompression de votre archive devra fournir tous les fichiers nécessaires **dans le répertoire courant où se situe l'archive**. Vous penserez à joindre tous les codes sources nécessaires.
- implémenter le type opaque permettant de manipuler, efficacement, un fichier `PNM`.
- être modulaire, i.e., nous nous attendons à trouver des fonctions/procédures en plus de ce qui vous est déjà fourni. Si nécessaire, vous pouvez ajouter des headers/modules mais, dans ce cas, vous devrez modifier le `Makefile` pour permettre la compilation de votre projet (nous ne le ferons pas pour vous).
- être parfaitement documenté. Vous veillerez à spécifier correctement chaque fonction/procédure/-structure de données que vous définirez.
- appliquer les principes de la programmation défensive (vérification des préconditions, vérification des mallocs, ...). Pensez à libérer la mémoire allouée en cours de programmation afin d'éviter les fuites de mémoire.

Afin de vous aider à tester votre librarie et votre programme, nous vous fournissons les fichiers `PNM` suivants :

- `totem.pgm`, 640 × 480 pixels.
- `poivron.pgm`, 255 × 255 pixels.
- `monalisa.pgm`, 250 × 350 pixels.
- `scs.pbm`, 136 × 82 pixels.
- `hippocampe.pbm`, 2225 × 3601 pixels.
- `washington.pbm`, 305 × 400 pixels.
- `escargot.ppm`, 256 × 256 pixels
- `antilope.ppm`, 512 × 512 pixels
- `arcenciel.ppm`, 300 × 300 pixels

En cas d'erreur renvoyée par une de vos fonctions, votre programme ne devra pas s'arrêter. Au contraire, il devra gérer au mieux l'erreur, par exemple en prévenant l'utilisateur de son erreur (cfr. l'exemple supra).

Toute question relative au projet peut être posée sur le forum de la page web du cours (eCampus)

Tout projet ne compilant pas se verra attribuer la note de 0/20.

Il est impératif de respecter **scrupuleusement** les consignes sous peine de se voir attribuer une note de 0/20 pour non respect de l'énoncé.