1 Fichiers

Pour lire/écrire dans un fichier, la librairie standard (<fstream>) fournit deux types de flux (entre autres) : (a) la classe std :: ifstream permet de lire un fichier ; (b) la classe std :: ofstream permet d'écrire dans un fichier. En écriture, deux modes sont possibles :

— std :: ios :: trunc : écraser le fichier s'il existe ou le créer,
— std :: ios :: app : écrire à la suite du fichier ou le créer.

Par exemple :

```
int toto = 234;
std::ofstream f; /* on instancie un flux en écriture */
f.open("yop.txt", std::ios::out | std::ios::trunc); /* ouverture du fichier */
f << "toto␣vaut" << "␣" << toto << "\n"; /* flux sortant << */
f.close(); /* fermeture du fichier */
```

Ou :

```
std::string input;
int toto;
std::ifstream f; /* on instancie un flux en lecture */
f.open("yop.txt", std::ios::in); /* ouverture du fichier */
f >> input >> toto; /* utilisation de l'opérateur flux entrant >> */
/* input vaut "toto vaut", toto vaut 234 */
f.close();
```

Par défaut ces modes sont ASCII (donc texte). Pour ouvrir en mode binaire, il faut rajouter std :: ios :: binary lors de l'ouverture du fichier :

```
std::ofstream f("yop.txt", std::ios::out | std::ios::trunc | std::ios::binary);
std::ifstream f("yop.txt", std::ios::in | std::ios::binary);
```

2 Lecture/Ecriture

2.1 Mode Texte

Pour lire et écrire dans un fichier en mode texte, le plus facile est d'utiliser les opérateurs de flux << et >> (voir doc).

2.2 Mode Binaire

Dans ce cas, il s'agit d'écrire ou de lire des zones de mémoire, en utilisant les méthodes write ou read. La syntaxe est :

```
int x = 10;
f.write((char*)&x, sizeof(int));
/* on écrit la valeur de x en binaire (ici 32bit) */
int y;
f.read((char*)&y, sizeof(int))
/* on lit le fichier sur 32bit et on affecte la valeur à y
  (cela suppose de connaître à l'avance le nombre
  d'octets stockés dans le fichier, ainsi que l'ordre
  de lecture */
```

Ainsi pour écrire (ou lire) un tableau de N doubles dans un fichier f on écrira :

```
f.write((char*)tab, sizeof(double)*N); /* écrit le tableau tab */
f.read((char*)tab, sizeof(double)*N); /* lit et sauve dans le tableau tab */
```

Pensez à réserver la mémoire pour accueillir les données dans le cas d'un read.
En vérité, il est possible d'utiliser la méthode `read` en ouvrant le fichier en mode texte.

3 Classe ImagePPM

Vous allez manipuler des images au format `ppm`.

Il s'agit d'un fichier binaire. Cependant, il commence par un en-tête (header) au format texte (ASCII) de la forme :

P6 \ n
W H \n
MaxVal \n

où : W est la largeur en pixels de l'image, H la hauteur en pixels de l'image et MaxVal est le nombre de degrés par canal maximum (ici ce sera toujours 255).

Ensuite, les données binaires (l'image elle-même) sont de la forme :

R G B R G B etc ... Il y a exactement $W * H$ triplets R G B et chaque R, G ou B est un entier non signé compris entre 0 et Maxval (donc pour nous un entier **unsigned char**). Il s'agit de la composante de Rouge (Red), Vert (Green) et Bleu (Blue) de chaque pixel dans l'image. Par exemple,

(0,0,0)	indique la couleur noire
(255,255,255)	indique le blanc
(255,0,0)	rouge pur
(0,255,0)	vert pur
(0,0,255)	bleu pur

1. Implémenter une classe ImagePPM, avec constructeurs, destructeurs etc... adéquats.
2. Implémenter une méthode de construction par fichier, et de sauvegarde dans un fichier.
3. Implémenter une méthode pour désaturer (transformer en niveaux de gris) une image.
4. Désaturer les images m.ppm, p.ppm, im.ppm
5. Implémenter une méthode appliquant le flou gaussien suivant :

$$\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}$$

6. Appliquer le flou gaussien aux images m.ppm, p.ppm, im.ppm.
7. Implémenter une image de Mandelbrot.