TP 8 : fichiers et images

Commme d'habitude, télécharger l'archive Semaine8.zip dans votre dossier Info111 et la désarchiver.

Exercice 1 (20 minutes)**.**

(1) Télécharger l'exemple du cours `fichier-ecriture.cpp` et l'enregistrer dans votre dossier Semaine8. Comme tous les exemples du cours, il se trouve dans la section `Documents` de la page web d'info 111.

(2) Ouvrir ce fichier avec un éditeur de texte (par exemple `gedit`) et essayer de deviner ce que fait ce programme.

(3) Exécuter ce programme. Rappel : pour exécuter un programme, il faut d'abord avoir compilé le fichier `.cpp` avec la commande `g++` puis lancer l'exécutable créé par la compilation, comme cela a été vu dans les TP précédents.

(4) Trouver le fichier `.txt` qui a été créé à l'exécution et l'ouvrir avec un éditeur de texte. Que contient-il ?

(5) Modifier le programme pour qu'à l'exécution il écrive un fichier nommé `essai.txt` contenant le texte « 17 fois 23 vaut » suivi de la valeur de ce nombre.

(6) Exécuter le programme puis ouvrir le fichier `essai.txt` afin de vérifier son contenu.

(7) Télécharger la dernière version des notes de cours (Semaine 7, « Fichiers »), et consulter la page « Lecture depuis le clavier ». Essayer l'exemple `cin.cpp`.

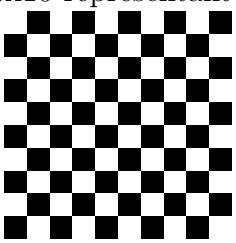
(8) En vous en inspirant, modifier votre programme pour que, à l'exécution, il demande à l'utilisateur d'entrer deux entiers, puis qu'il écrive un fichier nommé `multiplication.txt` contenant un texte similaire au fichier `essai.txt` de la question précédente, avec 17 et 23 remplacés respectivement par les deux entiers choisis par l'utilisateur.

(9) Exécuter le programme puis ouvrir le fichier `multiplication.txt` pour vérifier son contenu.

Exercice 2 (20 minutes)**.**

(1) Le fichier `smiley.pbm` contient l'image en noir et blanc du TD. Ouvrir ce fichier d'abord avec un éditeur de texte, puis avec un logiciel de vision d'images. Vous pouvez utiliser l'application `gwenview` sous Unix, ou l'application `irfanview` sous windows.

(2) Utiliser un éditeur de texte pour écrire à la main un *fichier* contenant une image `damier.pbm` au format PBM (*Portable Bit Map*, voir `http://fr.wikipedia.org/wiki/Portable_bitmap`) de taille 10x10 représentant un damier :



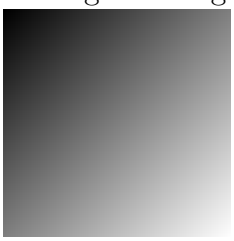
(3) Visualiser le résultat avec un logiciel de vision d'images

Exercice 3.
Implanter un **programme C++** `damier.cpp` qui lorsqu'on l'exécute écrit un fichier image `damier-automatique.pbm` comme le précédent, mais cette fois pour un damier 100x100.

Indication : Vous pouvez vous inspirer de `fichier-ecriture.cpp`. Commencer par un programme pour un damier 10x10 avant de passer à 100x100. *Si le fichier produit ne donne pas l'image attendue, ouvrez-le avec un éditeur de texte pour mieux comprendre ce qu'il se passe et aider au débogage.*

Exercice 4.

(1) Implanter un programme qui écrit un fichier contenant une image `degrade.pgm` au format PGM (*Portable Gray Map*, voir `http://fr.wikipedia.org/wiki/Portable_graymap`) de taille 255 par 255, avec un dégradé de gris :



(2) Répéter, avec une image de taille 100x100 puis 1000x1000.

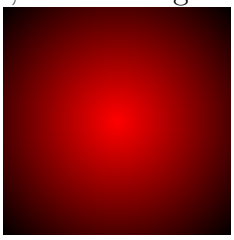
Exercice 5.
Implanter un programme qui lit un fichier contenant une image au format PGM (par exemple le fichier `image.pgm` fourni), et écrit un fichier contenant la même image en vidéo inverse (clair remplacé par sombre et réciproquement).

Indication : Implanter une fonction

```
/** Image en vidéo inverse
 * @param image1: le nom du fichier contenant l'image à lire
 * @param image2: le nom du fichier pour l'image à écrire
 **/
void videoInverse(string image1, string image2);
```

Exercice ♣ 6.

(1) Implanter un programme qui écrit un fichier contenant une image `degrade-circulaire.ppm` au format PPM de taille 255 par 255, avec un dégradé circulaire de rouge :



(2) Répéter, avec une image de taille 100x100 puis 1000x1000.