TP : Manipulation d'images

Objectifs : Ce TP a pour but de vous faire manipuler les fichiers texte, les enregistrements et les tableaux bi-dimensionnels.

Prérequis : Pour faire ce TP, il est nécessaire
- de connaître la notion de tableau ;
- de connaître la notion d'enregistrement ;
- de connaître la notion de fichiers texte ;
- de récupérer le fichier `manipulationImages.zip` (180711 octets).

1 Le format PGM

Q 1 . Récupérez le fichier `manipulationImages.zip` et décompressez-le dans votre dossier personnel. Vous devez obtenir un fichier nommé `lena.pgm` dont la taille est de 609781 octets. Cette image représente la célèbre photo de Lena Söderberg (1972), qui est l'une des images les plus utilisées pour tester des algorithmes de traitement d'image.

Q 2 . Ouvrez le fichier `lena.pgm` en utilisant un éditeur de texte (Kate par exemple). Ce fichier est-il un fichier texte ?

Ce fichier contient une image au format PGM. Le format PGM (Portable GrayMap) permet de coder chaque pixel d'une image en niveau de gris sur 8 bits (0 correspond au noir et 255 au blanc). La valeur de chacun des pixels est enregistrée dans le fichier au format ASCII. Les quatre premières lignes du fichier représentent l'entête. La première ligne "P2" permet d'indiquer qu'il s'agit bien d'un fichier au format PGM. Vient ensuite une ligne de commentaire qui commence par le symbole dièse pour indiquer le nom du créateur du fichier et une troisième ligne qui précise les dimensions de l'image (largeur puis hauteur). Enfin la quatrième ligne précise que les pixels sont codés sur 256 valeurs. Le reste du fichier donne les valeurs ASCII de chaque pixel ligne par ligne en commençant par le pixel supérieur gauche de l'image.

```
P2
# CREATOR: GC
512 330
255
141
141
139
...
```

Il est possible d'afficher une image en utilisant la commande display. Tapez `display lena.pgm` en ligne de commande pour visualiser l'image.

2 Le type IMAGE

Les images sont représentées en Pascal par le type IMAGE défini ci-dessous

```
const
   TAILLEMAX = 1024; // Taille maximale d'une image

type
   BYTE        = 0..255; // 0: noir, 255: blanc
   IMAGE = record
         largeur : CARDINAL; // largeur de l'image
         hauteur : CARDINAL; // hauteur de l'image
    // Tableau bi-dimensionnel contenant les pixels de l'image
         PIXEL   : ARRAY [1..TAILLEMAX, 1..TAILLEMAX] of BYTE;
   end;
```

Pour les questions qui suivent, nous travaillerons avec des images dont la taille maximale (hauteur ou largeur) ne dépassera pas 1024 pixels.

3 Lecture d'une image

Q 3 . Ecrire une procédure procedure *ChargerImage* dont la spécification est la suivante :

```
// ChargerImage(nomfichier, img)
// lit l'ensemble des pixels du fichier nomfichier au format PGM et les
// enregistre dans img

procedure ChargerImage(??? nomFichier :STRING; ??? img:IMAGE);
```

4 Enregistrement d'une image

Q 4 . Ecrire une procédure procedure *EnregistrerImage* dont la spécification est la suivante :

```
// EnregistrerImage(nomfichier, img)
// enregistre l'ensemble des pixels contenus dans img dans le fichier
// nomFichier au format PGM

procedure EnregistrerImage(??? nomFichier :STRING; ??? img:IMAGE);
```

Q 5 . Charger l'image lena.pgm avec la procdédure *ChargerImage* et enregistrez là avec le nom lena2.pgm avec la procédure *EnregistrerImage*. Vérifiez que vous obtenez la même image.

5 Dessiner un monochrome

Q 6 . Ecrire une fonction *ImageMonochrome* dont la spécification est la suivante :

```
// ImageMonochrome(niveauDeGris, largeur, hauteur)
// crée une image de taille largeur × hauteur de
// couleur unie définie par niveauDeGris
// Par exemple ImageMonochrome(127,100,200) donne une image
// de dimension 100×200 dont tous les pixels ont
// un niveau de gris valant 127

function ImageMonochrome(??? niveauDeGris:BYTE; ??? largeur:CARDINAL;
                         ??? hauteur:CARDINAL):IMAGE;
```

Q 7 . Enregistrez votre image sous le nom monochrome.pgm et visualisez le résultat.

6 Dessiner un carré

Q 8 . Ecrire la fonction *Carre* dont la spécification est la suivante :

```
// Carre(largeur, hauteur, cote)
// produit une image de taille largeur × hauteur
// avec un fond blanc et un carre noir centré
// sur l'image dont le côté a pour dimension cote

function Carre(??? largeur:CARDINAL; ??? hauteur:CARDINAL;
               ??? cote:CARDINAL):IMAGE;
```

Q 9 . Enregistrez votre image sous le nom carre.pgm et visualisez le résultat.

7 Binarisation d'une image

La binarisation d'une image permet d'obtenir une image constituée de pixels blancs ou noirs à partir d'une image en niveaux de gris, après avoir défini un seuil. Les pixels dont le niveau de gris est inférieur au seuil deviennent noirs et les autres blancs.

Q 10 . Ecrire la fonction *Binarisation* dont la spécification est la suivante :

```
// Binarisation(img, seuil)
// produit une IMAGE binarisée à partir de img: les pixels
// dont le niveau de gris est inférieur à seuil deviennent noirs
// et les autres blancs.

function Binarisation (??? img : IMAGE; ??? seuil:BYTE):IMAGE;
```

FIG. 1 – Exemple d'image avant (gauche) et après binarisation (droite).

Q 11 . Enregistrez l'image binariséee sous le nom binarisation.pgm et visualisez le résultat. Testez différents seuils.

8 Miroir vertical

Q 12 . Ecrire la fonction *MiroirVertical* dont la spécification est la suivante :

```
// MiroirVertical(img)
// produit une image qui correspondrait à l'image visible dans
// un miroir placé à côté de l'image

function MiroirVertical(??? img:IMAGE):IMAGE;
```

FIG. 2 – Exemple d'effet miroir vertical.

Q 13 . Enregistrez l'image obtenue sous le nom miroir.pgm et visualisez le résultat.

9 Rotation

Q 14 . Ecrire la fonction *Rotation90degres* dont la spécification est la suivante :

```
// Rotation90degres(img, sensTrigo)
// produit une image pivotée à 90 degrés dans le sens trigo
// ou dans le sens horaire

function Rotation90degres(??? img:IMAGE; ??? sensTrigo:BOOLEAN):IMAGE;
```

Q 15 . Enregistrez l'image pivotée sous le nom rotation.pgm et visualisez le résultat.

10 Création de l'histogramme

L'histogramme d'une image permet de visualiser la distribution des pixels dans une image suivant leur niveau de gris. Il se représente sous forme de diagramme à barres avec en abscisse le niveau de gris entre 0 et 255 et en ordonnée le nombre d'apparition de ce niveau dans l'image.

Pour traiter cette partie, nous allons travailler avec le type HISTOGRAMME défini ci-dessous :

```
type
  HISTOGRAMME = ARRAY [0..255] of CARDINAL;
```

Q 16 . En utilisant Gimp, visualisez l'histogramme de l'image lena.pgm : Menu `Dialogues` puis `Histogramme`.

Q 17 . Ecrire la fonction *CalculHistogramme* dont la spécification est la suivante :

```
// CalculHistogramme(img)
// produit l'histogramme de l'image img
function CalculHistogramme(const img:IMAGE): HISTOGRAMME;
```

Q 18 . Affichez les valeurs contenues dans le tableau HISTOGRAMME. Calculer la plus grande valeur possible du tableau en fonction des dimensions de l'image.

Q 19 . Pour produire une image semblable à celle produite par Gimp, il va falloir normaliser les valeurs de l'histogramme. La normalisation de l'histogramme consiste dans un premier temps à déterminer la valeur maximale de l'histogramme (max) puis à appliquer un changement d'échelle entre l'intervalle [0, max] et un autre intervalle plus petit ([0, 100] par exemple) pour que l'image ait une hauteur raisonnable.

Ecrire la procédure *normaliserHistogramme* dont la spécification est la suivante :

```
// normaliserHistogramme(histo, h)
// normalise l'histogramme en faisant un changement d'échelle de [0, max]
// max étant la valeur maximale de l'histogramme à [0, h]

procedure normaliserHistogramme(??? histo:HISTOGRAMME; ??? h:CARDINAL);
```

Q 20 . Ecrire la fonction *CalculImageHistogramme* dont la spécification est la suivante :

```
// CalculImageHistogramme(img)
// calcule l'histogramme de l'image img et renvoie le
// résultat sous forme de diagramme à barres. L'image produite
// a une largeur égale à 256 pixels et une hauteur passée en 2e paramètre

function CalculImageHistogramme(??? img: IMAGE; ??? hauteur:CARDINAL):IMAGE;
```

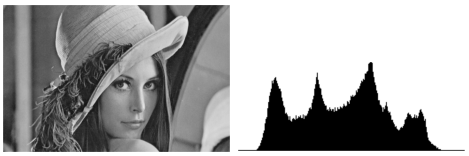

FIG. 3 – Exemple d'histogramme d'une image.

Q 21 . Enregistrez l'histogramme sous le nom histogramme.pgm et comparez votre résultat avec celui donné par Gimp.