Séance 14 : Jeudi 11/04/2019

On souhaite développer un logiciel permettant d'effectuer des traitements simples sur les images : extraction de valeurs caractéristiques, seuillage, miroir horizontal, addition.


image initiale

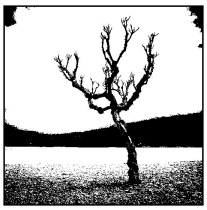
image seuillée


miroir image


initiale + miroir

Une image est constituée d'une grille (tableau) de pixels ayant une intensité lumineuse (niveau de gris). Le niveau de gris d'un pixel est une valeur entière comprise entre 0 (noir) et 255 (blanc).

1. Afin de pourvoir gérer des images de tailles variables, on définira une taille maximale MAX_X et MAX_Y pour les tableaux utilisés. Définir en langage C / C++ deux constantes ayant pour valeur 256.
2. Écrire en langage C / C++ la structure de données permettant de stocker une image. Cette image sera identifiée par sa taille effective en nombre de pixels (taille_x et taille_y) et un tableau contenant les intensités lumineuses de chacun des pixels.
3. Écrire en langage C / C++ une fonction qui permet de remplir la structure image.
4. Écrire en langage C / C++ un sous-programme qui permet d'afficher les valeurs des différents pixels de l'image.
5. Ecrire en langage C / C++ un sous-programme permettant d'extraire, en un seul parcours de l'image :
    - le niveau de gris minimum de l'image,
    - le niveau de gris maximum de l'image,
    - le niveau de gris moyen de l'image.
6. L'opération dite de « seuillage simple » consiste à mettre à zéro tous les pixels ayant un niveau de gris inférieur à une certaine valeur (appelée seuil) et à mettre à la valeur maximale les pixels ayant une valeur supérieure à ce seuil. Ainsi le résultat du seuillage est une image binaire ne contenant que des pixels noirs et blancs. Écrire en langage C / C++ un sous-programme permettant d'effectuer le seuillage d'une image. Le seuil choisi par l'utilisateur sera passé en paramètre.
7. Écrire en langage C / C++ un sous-programme permettant d'effectuer la symétrie verticale d'une image (miroir). Le résultat sera stocké dans une nouvelle image qui sera retournée au programme principal.
8. Écrire en langage C / C++ un sous-programme permettant de retourner une nouvelle image, calculée comme étant la somme de deux images passées en paramètres. Lorsque la somme des intensités lumineuses des deux pixels ajoutés est supérieure à 255, on la fixera à cette valeur limite.
9. Écrire en langage C / C++ un sous-programme permettant de proposer le menu suivant

```
= = MENU= =
0-  QUITTER
1- Afficher intensités min, max et moyenne
2- Seuiller l'image
3- Symétrie de l'image
4- Somme de deux images
```

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;
const int MAX_X = 256 ;
const int MAX_Y = 256 ;

struct image
{
    int taille_x, taille_y ;
    int tabim[MAX_X][MAX_Y];
};

// demander taille_x et taille_y à l'utilisateur en vérifiant qu'on a une
valeur
//comprise entre 0 et 255
// mettre une intensité lumineuse de manière aléatoire dans chaque case

struct image remplir_image_fct (){

    struct image im;
    int i,j;
    do
    {
        cout<<"Donnez les dimensions de l'image"<<endl;
        cin>>im.taille_x>>im.taille_y;
    } while (im.taille_x<=0 || im.taille_x>256 || im.taille_y <=0 ||
im.taille_y>256);

    for (i=0;i<im.taille_x;i++)
    {
        for (j=0;j<im.taille_y;j++)
        {
            im.tabim[i][j] = rand() % 256 ;
        }
    }

    return im;

}
void remplir_image_proc (struct image &im){
    int i,j;
    do
    {
        cout<<"Donnez les dimensions de l'image"<<endl;
        cin>>im.taille_x>>im.taille_y;
    } while (im.taille_x<=0 || im.taille_x>256 || im.taille_y <=0 ||
im.taille_y>256);

    for (i=0;i<im.taille_x;i++)
    {
        for (j=0;j<im.taille_y;j++)
        {
            im.tabim[i][j] = rand() % 256 ;
        }
    }
}
void afficher_image (struct image im){
    int i,j;
    for (i=0;i<im.taille_x;i++)
    {
        for (j=0;j<im.taille_y;j++)
        {
            cout<<im.tabim[i][j]<<" " ;
        }
        cout<<endl;
```

```
    }
    cout<<endl;
}
void extraction (struct image im, int &ng_min, int & ng_max, int & ng_moy){
    int i,j;
    ng_min = 255;
    ng_max = 0 ;
    ng_moy = 0;
    for (i=0;i<im.taille_x;i++)
    {
        for (j=0;j<im.taille_y;j++)
        {
            if (im.tabim[i][j]<ng_min) ng_min = im.tabim[i][j];
            if (im.tabim[i][j]>ng_max) ng_max = im.tabim[i][j];
            ng_moy += im.tabim[i][j] ;
        }
    }
    ng_moy /= (im.taille_x * im.taille_y);
}
void seuillage (struct image im, struct image & ims, int seuil){
    int i,j;
    ims = im ; // pour recopier les valeurs des taille_x et taille_y
    for (i=0;i<im.taille_x;i++)
    {
        for (j=0;j<im.taille_y;j++)
        {
            if (im.tabim[i][j]>seuil) ims.tabim[i][j] = 255;
                else ims.tabim[i][j] = 0 ;
        }
    }
}
struct image miroir (struct image im){
    struct image mir ;
    int i,j;
    mir = im;
    for (i=0;i<im.taille_x;i++)
    {
        for (j=0;j<im.taille_y;j++)
        {
            mir.tabim[i][j] = im.tabim[i][im.taille_y-j-1];
        }
    }
    return mir ;
}
struct image somme (struct image im1, struct image im2){
    struct image imsom;
    int i,j, som;
    imsom = im1;
    for (i=0;i<im1.taille_x;i++)
    {
        for (j=0;j<im1.taille_y;j++)
        {
           som = im1.tabim[i][j] +  im2.tabim[i][j];
           if (som>255) imsom.tabim[i][j] = 255;
           else imsom.tabim[i][j] = som ;
        }
    }
    return imsom ;
}
int menu(){
    int choix ;
    cout<<"Vous avez les choix suivants : "<<endl ;
    cout<<"0 : Quitter "<<endl;
    cout<<"1 : Extraire les valeurs caracteristiques de l'image"<<endl;
    cout<<"2 : Seuiller l'image"<<endl;
    cout<<"3 : Effectuer la symetrie verticale de l'image"<<endl;
```

```
    cout<<"4 : Sommer deux images"<<endl;

    do
    {
        cout<<"Quel est votre choix ?"<<endl;
        cin>>choix;
    } while (choix<0 || choix>4);
    return choix ;
}
int main (void)
{
    struct image im1, im_seuil, im_miroir, im_som;
    int ngm,ngM,ngmoy, seuil, chx;
    srand(time(NULL));
    do
    {
        chx = menu();
        switch (chx)
        {
            case 1 : im1 = remplir_image_fct();
                     afficher_image(im1);
                     extraction(im1,ngm,ngM,ngmoy);
                     cout<<"Le niveau de gris le plus petit est :
"<<ngm<<endl;
                     cout<<"Le niveau de gris le plus grand est :
"<<ngM<<endl;
                     cout<<"Le niveau de gris moyen est : "<<ngmoy<<endl;
                break;
            case 2 : im1 = remplir_image_fct();
                     afficher_image(im1);
                     seuillage(im1,im_seuil, 128);
                     afficher_image(im_seuil);
                break;
            case 3 : im1 = remplir_image_fct();
                     afficher_image(im1);
                     im_miroir = miroir(im1);
                     afficher_image(im_miroir);
                break;
            case 4 : im1 = remplir_image_fct();
                     afficher_image(im1);
                     im_som = somme(im1,im1);
                     afficher_image(im_som);
                break;
            default : cout <<"Au revoir"<<endl;
        }
    } while (chx != 0);
    return 0;
}
```

Des tableaux de structures

Une fleur est définie par

- ✓ son nom,
- ✓ sa couleur,
- ✓ **un tableau** comparatif de son prix chez 3 commerçants sélectionnés.

Une liste de fleurs est définie par

- ✓ le nombre de fleurs qu'elle contient,
- ✓ un tableau de fleurs.

1. Définir en langage C/C++ une constante CHMAX ayant pour valeur 50 qui sera utilisée comme taille maximale des chaines de caractères.

2. Définir en langage C/C++ les structures de données `fleur` et `liste_fleurs` permettant de stocker toutes les informations concernant une fleur et une liste de fleurs.

3. Écrire en langage C/C++ un sous-programme `saisir_fleur` permettant de saisir toutes les informations relatives à une fleur. Attention on recommencera la saisie tant que les données de prix ne sont pas strictement positives.

4. Écrire en langage C/C++ un sous-programme `afficher_fleur` qui permet d'afficher les caractéristiques d'une fleur.

5. Écrire en langage C/C++ un sous-programme `saisir_liste_fleurs` permettant de saisir toutes les informations relatives à une liste de fleurs. On utilisera pour cela le sous-programme de la question 3.

6. Afin d'effectuer des comparaisons tarifaires entre les 3 commerçants sélectionnés, on souhaite connaître pour une liste de fleurs, le prix total de toutes les fleurs chez chacun des 3 commerçants. Écrire en langage C/C++ **un seul** sous-programme `prix_liste_fleurs` permettant d'extraire ces 3 informations. On s'assurera de ne parcourir qu'une seule fois le tableau de fleurs.

7. Écrire en langage C/C++ un sous-programme `affiche_fleurs_rouges` permettant d'afficher le nom de toutes les fleurs de couleur rouge de la liste de fleurs.

8. Écrire en langage C/C++ un sous-programme `ajoute_une_fleur` permettant d'ajouter une fleur au tableau de fleur s'il reste de la place dans le tableau.

9. Écrire en langage C/C++ le programme principal permettant de remplir un tableau avec autant de fleurs que l'utilisateur le voudra, d'afficher le nom de toutes les fleurs rouges de cette liste et d'afficher le numéro du commerçant le moins cher des 3 pour cette liste.

```
#include <iostream>
#include <string.h>
using namespace std;
const int CHMAX = 64 ;
const int MAXFLEUR = 64 ;

struct fleur
{
    char nom[CHMAX];
    char couleur[CHMAX];
    float prix[3];
};

struct bouquet
{
    int nbfleurs;
    struct fleur tabfleur[MAXFLEUR];
};

void saisir_fleur( struct fleur &f){
    int i;
    cout<<"Donnez le nom de la fleur : "<<endl;
    cin>>f.nom ;
    cout<<"Donnez la couleur de la fleur : "<<endl;
```

```
    cin>>f.couleur ;
    for (i=0;i<=2;i++)
    {
        do
        {
            cout<<"Donnez le prix de la fleur chez le commercant "<<i<<endl;
            cin>>f.prix[i] ;
        } while (f.prix[i]<=0);
    }
}
void afficher_fleur( struct fleur f){
    int i;
    cout<<"Le nom de la fleur : ";
    cout<<f.nom<<endl ;
    cout<<"La couleur de la fleur : ";
    cout<<f.couleur<<endl ;
    for (i=0;i<=2;i++)
    {
        cout<<"DLe prix de la fleur chez le commercant "<<i<<" est ";
        cout<<f.prix[i]<<endl ;
    }
}
void saisir_bouquet(struct bouquet &b){
    int i;
    cout<<"Nombre de fleurs dans le bouquet"<<endl;
    cin>>b.nbfleurs;
    for (i=0;i<b.nbfleurs;i++)
    {
        saisir_fleur(b.tabfleur[i]);
    }
}
void prix_bouquet(struct bouquet b, float &pcom0, float & pcom1, float & pcom2){
    pcom1=0;
    pcom2=0;
    pcom0=0;
    int i;
    for (i=0;i<b.nbfleurs;i++)
    {
        pcom0+=b.tabfleur[i].prix[0];
        pcom1+=b.tabfleur[i].prix[1];
        pcom2+=b.tabfleur[i].prix[2];

    }
}
void affiche_fleurs_rouges(struct bouquet b){
    int i;
    for (i=0;i<b.nbfleurs;i++)
    {
        if (strcmp(b.tabfleur[i].couleur, "rouge")==0)
            cout<<b.tabfleur[i].nom<<endl;
    }
}
void ajoute_une_fleur(struct bouquet &b){
    if (b.nbfleurs<MAXFLEUR)
    {
       saisir_fleur(b.tabfleur[b.nbfleurs]);
       b.nbfleurs++;
    }
    else
        cout<<"Plus de place dans le tableau !"<<endl;

}
int main (void)
{
    struct bouquet b1;
    char rep ;
```

```
    b1.nbfleurs = 0;
    float p0,p1,p2;
    do
    {
        ajoute_une_fleur(b1);
        cout<<"encore une fleur ?"<<endl;
        cin>>rep;
    } while(rep=='o');

    affiche_fleurs_rouges(b1);
    prix_bouquet(b1,p0,p1,p2);
    cout<<"Prix du bouquet chez le commercant 0 : "<<p0<<endl;
    cout<<"Prix du bouquet chez le commercant 1 : "<<p1<<endl;
    cout<<"Prix du bouquet chez le commercant 2 : "<<p2<<endl;
    if (p0<p1 && p0<p2) cout<<"Le commercant le moins cher est le 0 avec "<<p0;
    if (p1<p0 && p1<p2) cout<<"Le commercant le moins cher est le 1 avec "<<p1;
    if (p2<p0 && p2<p1) cout<<"Le commercant le moins cher est le 2 avec "<<p2;
    return 0;
}
```