

LIFAP1 – TD 9 : Les structures

Objectifs : Manipulation des structures
Application aux tableaux

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



image initiale

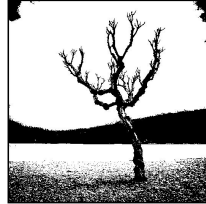


image seuillée



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 pouvoir gérer des images de tailles variables, on définira une taille maximale MAX_X et MAX_Y pour les tableaux utilisés. Définir deux constantes ayant pour valeur 256.

```
const int MAX_X = 256;  
const int MAX_Y = 256;
```

2. Écrire en langage algorithmique puis 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.

```
struct image  
{  
    int taille_x;  
    int taille_y;  
    int tab_img[MAX_X][MAX_Y];  
};
```

Structure image
Taille_x, taille_y : entiers
Tab_img[MAX_X][MAX_Y] : entiers
Fin structure image

3. Écrire en langage algorithmique puis en langage C / C++ une fonction qui permet de remplir la structure image.

```
struct image saisir_image()  
{  
    int i,j;  
    struct image im;  
    cout<<"quelle est la taille de l'image ?"<<endl;  
    cin>>im.taille_x>>im.taille_y;  
    for (i=0;i<im.taille_x;i++)  
    {  
        for (j=0;j<im.taille_y;j++)  
        {  
            cout<<"niveau de gris du pixel"<<i<<","<<j<<endl;  
            cin>>im.tab_img[i][j];  
        }  
    }  
    return im;  
}
```

4. Écrire en langage C / C++ un sous-programme qui permet d'afficher les valeurs des différents pixels de l'image.

```
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.tab_img[i][j];
        }
        cout<<endl;
    }
}
```

5. Écrire 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.

Void extraire (struct image im, int &min, int &max, int &moy)

```
{
    int i,j,som ;
    min=255 ;
    max=0 ;
    som=0 ;
    for (i=0 ;i<im.taille_x ;i++)
    {
        for (j=0 ;j<im.taille_y ;j++)
        {
            som+=im.tab_img[i][j] ;
            if (im.tab_img[i][j] > max)
                max= im.tab_img[i][j] ;
            if (im.tab_img[i][j] < min)
                min = im.tab_img[i][j] ;
        }
    }
    moy= som / (im.taille_x * im.taille_y) ;
}
```

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.

```
void seuillage (struct image &im, int seuil)
{
    int i,j;
    for (i=0;i<im.taille_x;i++)
    {
        for (j=0;j<im.taille_y;j++)
        {
            if (im.tab_img[i][j] > seuil)
                im.tab_img[i][j]=255;
            else
                im.tab_img[i][j]=0;
        }
    }
}
```

```

        im.tab_img[i][j]=0;
    }
}

```

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.

```

// version en modifiant l'image d'entrée
void miroir (struct image &im)
{
    int i,j;
    int ng;
    for (i=0;i<im.taille_x;i++)
    {
        for (j=0;j<im.taille_y / 2;j++)
        {
            ng = im.tab_img[i][j];
            im.tab_img[i][j]=im.tab_img[i][im.taille_y - j - 1];
            im.tab_img[i][im.taille_y - j - 1] = ng;
        }
    }
}

// version créant une image miroir
void miroir (struct image im, struct image &miroir)
{
    int i,j;

    miroir = im ;

    for (i=0;i<im.taille_x;i++)
    {
        for (j=0;j<im.taille_y;j++)
        {
            miroir.tab_img[i][j]=im.tab_img[i][im.taille_y - j - 1];
        }
    }
}

```

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.

```

struct image somme (struct image im1, struct image im2)
{
    int i,j;
    struct image somme;
    somme = im1;
    for (i=0;i<im1.taille_x;i++)
    {
        for (j=0;j<im1.taille_y;j++)
        {
            if (somme.tab_img[i][j] + im2.tab_img[i][j] >= 255)
                somme.tab_img[i][j]=255;
            else
                somme.tab_img[i][j]+= im2.tab_img[i][j];
        }
    }
}

```

```

    }
}
return somme;
}

```

9. Nous souhaitons maintenant construire un dessin animé. Nous allons pour cela gérer plusieurs images. Définir la structure permettant de stocker au plus NB_IMAGE, puis les sous-programmes permettant d'ajouter et de supprimer une image de notre dessin animé.

```

struct dessin_anime
{
    int nb_images;
    struct image T[NB_IMAGE];
};

void ajout_image (struct dessin_anime &DA, struct image im)
{
    DA.T[DA.nb_images] = im ;
    DA.nb_images ++ ;
}

void retire_image (struct dessin_anime &DA, int numero_im)
{
    int i ;
    for (i = numero_im ; i < DA.nb_images - 1 ; i++)
    {
        DA.T[i] = DA.T[i+1] ;
    }
    DA.nb_images -- ;
}

```

Pour s'entraîner

10. Écrire en langage C / C++ un sous-programme permettant de proposer le menu suivant à l'utilisateur :

<p style="text-align: center;">== MENU ==</p> <p>0- Saisir une image</p> <p>1- Afficher intensités min, max et moyenne</p> <p>2- Seuiller l'image</p> <p>3- Symétrie de l'image</p> <p>4- Somme de deux images</p> <p>5- QUITTER</p>
--

```

void affiche_menu()
{
    cout<<"LOGICIEL DE TRAITEMENT D'IMAGES"<<endl;
    cout<<"0 - Saisir une image"<<endl;
    cout<<"1 - Extraire des valeurs caractéristiques"<<endl;
    cout<<"2 - Seuiller l'image"<<endl;
    cout<<"3 - Miroir de l'image"<<endl;
}

```

```

    cout<<"4 - Sommer deux images"<<endl;
    cout<<"5 - Quitter"<<endl;
}
void menu ()
{
    int choix;
    struct image ima1,ima2, imas;
    int mini, maxi, moy, seuil;
    do
    {
        affiche_menu();
        cout<<"CHOIX ? : "<<endl;
        cin>>choix;
        switch (choix)
        {
            case 0 :
                ima1=saisir_image();
                afficher_image(ima1);
                break;
            case 1 :
                extraire(ima1,mini,maxi,moy);
                cout<<"min : "<<mini<<"max : "<<maxi<<"moyenne : "<<moy<<endl;
                break;
            case 2 :
                cout<<"quel seuil voulez vous appliquer"<<endl;
                cin>>seuil;
                seuillage(ima1,seuil);
                afficher_image(ima1);
                break;
            case 3 :
                miroir(ima1);
                afficher_image(ima1);
                break;
            case 4 :
                ima2=saisir_image();
                imas=somme(ima1,ima2);
                afficher_image(imas);
                break;

            default : cout<<"au revoir"<<endl;
        }
    }while (choix !=5);
}

```