IMT Lille Douai
École Mines-Télécom
IMT-Université de Lille

Luc Fabresse
luc.fabresse@imt-lille-douai.fr
CERI Numérique

Un peu dessin

Dans ce TP, nous allons utiliser la bibliothèque `libBMP` (très simplifiée) pour lire et écrire des images au format *bitmap* (BMP). Cette bibliothèque définit [1] :

- les types de données suivants : `byte`, `boolean`, `Color` et `Image`
- les fonctions suivantes :
  - `Image *CreateImage(int width,int height)`
    Retourne un pointeur sur une structure image dont la taille allouée est `width`*`height`
  - `void DestroyImage(Image *image)`
    Libère l'espace mémoire de `image`
  - `void SaveImage(char *filename, Image *image)`
    Sauvegarde le contenu de la structure image dans un fichier de nom `filename`
  - `void SetPixel(Image *image,int x,int y,Color c)`
    Fixe la couleur du pixel (x,y) à c
  - `Color GetPixel(Image *image,int x,int y)`
  - `int GetWidth(Image *image)`
  - `int GetHeight(Image *image)`

Voici un exemple d'utilisation de cette bibliothèque :

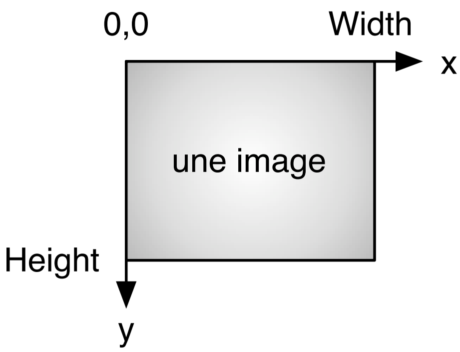
```c
#include "libBMP.h"

int main(int argc, char *argv[]) {
  Color blue = {0,0,255};
  int i,j;
  Image *image = CreateImage(100,100);

  for(j=0;j<GetHeight(image);j++)
    for(i=0;i<GetWidth(image);i++)
      if((i+j)%2)
        SetPixel(image,i,j,blue);

  SaveImage("test.bmp",image);
  DestroyImage(image);

  return 0;
}
```




```
$ gcc -Wall -std=c99 -c libBMP.c
$ gcc -Wall -std=c99 testLibBMP.c libBMP.o -o testLibBMP
$ ./testLibBMP.exe
```

1. Écrire la fonction : `void effacer(Image *image, Color c)`
   qui met tous les pixels de l'image à la couleur `c`.
2. Écrire la fonction :

   `void traceRectangle(Image *image,int xi,int yi,int largeur,int hauteur,Color c)`

   qui trace un rectangle à une position donnée (`xi`,`yi`), de taille `(largeur,hauteur)` avec la couleur `c`.
3. Écrire la fonction :
   `void ligneCartesienne(Image *image,int xi,int yi,int xf,int yf,Color c)`
   qui trace une ligne entre les points (`xi`,`yi`) et (`xf`,`yf`). On suppose que $xi < xf$. Rappel : l'équation cartésienne d'une droite est $y = ax + b$ où $a = \frac{yf-yi}{xf-xi}$
   La fonction `ligneCartesienne` est simple mais peu efficace car utilise des opérations sur des réels (`float`). Dans la suite, utilisez la fonction `ligneBresenham` (fournie par `libBMP`) qui

1. Voir le fichier `libBMP.h` pour plus de détails

implémente un algorithme est plus complexe (proposé par Bresenham) n'utilisant que des opérations sur des entiers.

4. Écrire la fonction :

   `void remplissage(Image *image,int x,int y,Color c,Color lim)`

   qui « colorie » avec la couleur c tous les pixels contenus dans une zone de l'image délimitée par la couleur lim. Le pixel (x, y) est à l'intérieur de la zone à « colorier ».

   Aide : Faire un algorithme récursif (bien plus simple) qui colorie le pixel (x, y) si besoin et remplit ensuite les quatre proches voisins de (x, y) à savoir : $(x+1, y)$, $(x-1, y)$, $(x, y+1)$, $(x, y-1)$. Attention, une telle version récursive peut provoquer des dépassements de capacité de la pile si la région à colorier est trop importante.

5. Étant donné la fonction (donnée dans la bibliothèque `libBMP`) :

   `void traceEllipse(Image *image,int xi,int yi,int a,int b,Color c)`

   Écrire les procédures suivantes :

   `void traceEllipsePleine(Image *image,int xi,int yi,int a,int b,Color c)`

   qui trace une ellipse remplie avec la couleur `c`.

   `void traceCercle(Image *image,int xi,int yi,int rayon,Color c,boolean remplir)`

   qui trace un cercle éventuellement rempli avec la couleur `c`.

6. Écrire une fonction permettant d'obtenir le dessin suivant :

La fractale Mandelbrot

La fractale de Mandelbrot représente un ensemble de points c du plan complexe pour lesquels la suite récurrente définie par : $z_{n+1} = z_n^2 + c$ et $z_0 = 0$, ne tend pas vers l'infini (en module). Pour déterminer si un point c est dans l'ensemble de Mandelbrot, il faut calculer les termes de la suite précédente. Deux cas sont alors possibles :

— Le module de z_n devient supérieur à 2. Dans ce cas, il peut être démontré que c n'appartient pas à l'ensemble de Mandelbrot.

— Le module de z_n reste inférieur à 2 et il faut interrompre le calcul après un certains nombre d'itérations fixé par le programme (`borne`). Dans ce cas, le programme considérera que c appartient à l'ensemble de Mandelbrot même si une `borne` plus grande aurait pu permettre de rejeter c. Il en résulte que l'image affichée n'est qu'une approximation.

Rappels sur les nombres complexes :

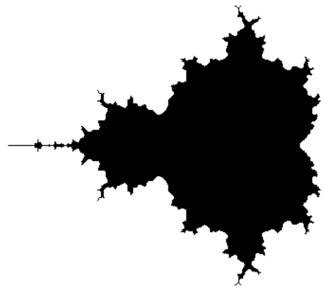
— définition : $z = a + ib$, a est la partie réelle et b la partie imaginaire de z

— addition : $(a_1 + i * b_1) + (a_2 + i * b_2) = (a_1 + a_2) + i * (b_1 + b_2)$

— multiplication : $(a_1 + i * b_1) * (a_2 + i * b_2) = (a_1 * a_2 - b_1 * b_2) + i * (a_1 * b_2 + b_1 * a_2)$

— module : $(a + i * b) = \sqrt{a^2 + b^2}$

1. Ecrire un programme qui parcourt les points du plan complexe contenus dans le rectangle $(-2.2, 1.5)$ et $(0.8, -1.5)$ avec une précision de 0.015 et les représente dans une image de taille 200×200. Le point complexe $(-2.2, 1.5)$ sera représenté par le pixel $(0, 0)$ en haut à gauche de l'image. Un pixel sera blanc s'il représente un point qui n'appartient pas à l'ensemble de Mandelbrot et noir dans le cas contraire.



2. Ajouter une fonction de coloration de la fractale. Au lieu d'attribuer la couleur blanche aux points qui n'appartiennent pas à l'ensemble de Mandelbrot, sa couleur sera déterminée en fonction de l'itération à partir de laquelle il a été rejeté. La bibliothèque `math.h` fournit tous les fonctions mathématique dont vous pourriez avoir besoin : `pow`, `log`, ...