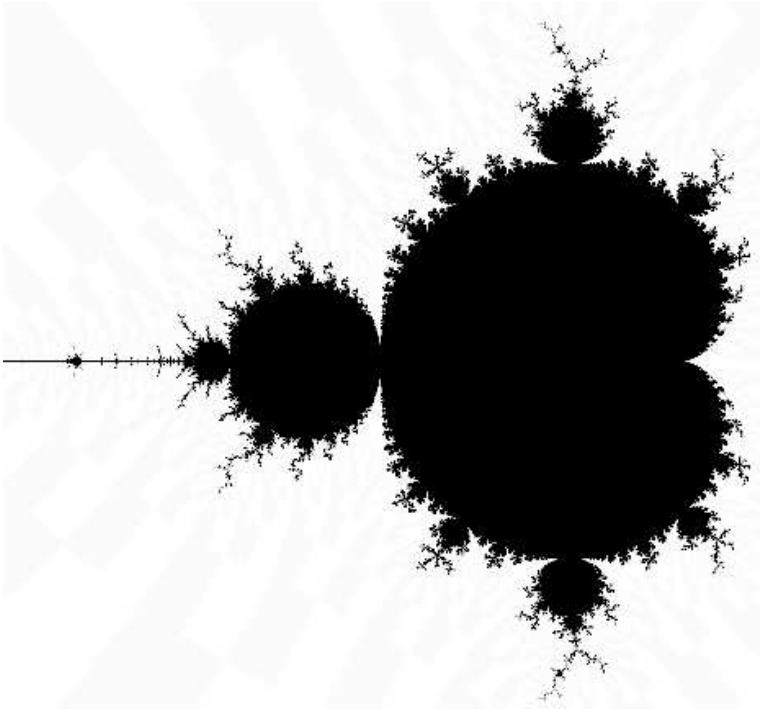
Université de Limoges
Faculté des Sciences
Master Procédés et Matériaux

Année 2005-2006

Travaux Dirigés - Langage C
Série 2

Le pays de Tor'Bled-Nam

Imaginons que nous avons fait un long voyage en direction d'un monde lointain. Nous appellerons ce monde Tor'Bled-Nam. Notre dispositif de détection à distance a capté un signal qui est à présent sur la feuille devant vous. L'image est centrée et nous voyons la chose suivante :



De quoi peut-il s'agir ? Est-ce un insecte aux allures bizarres ? Qu'est-ce que cette contrée étrange sur laquelle nous sommes tombés ? Ce monde n'est rien d'autre qu'un échantillon de mathématique abstraite : l'ensemble connu sous le nom d'**ensemble de Mandelbrot**. Il est sans aucun doute complexe ; et pourtant il est engendré par une règle d'une remarquable simplicité.

On considère la suite récurrente définie par :

$$z_{n+1} = z_n^2 + c$$

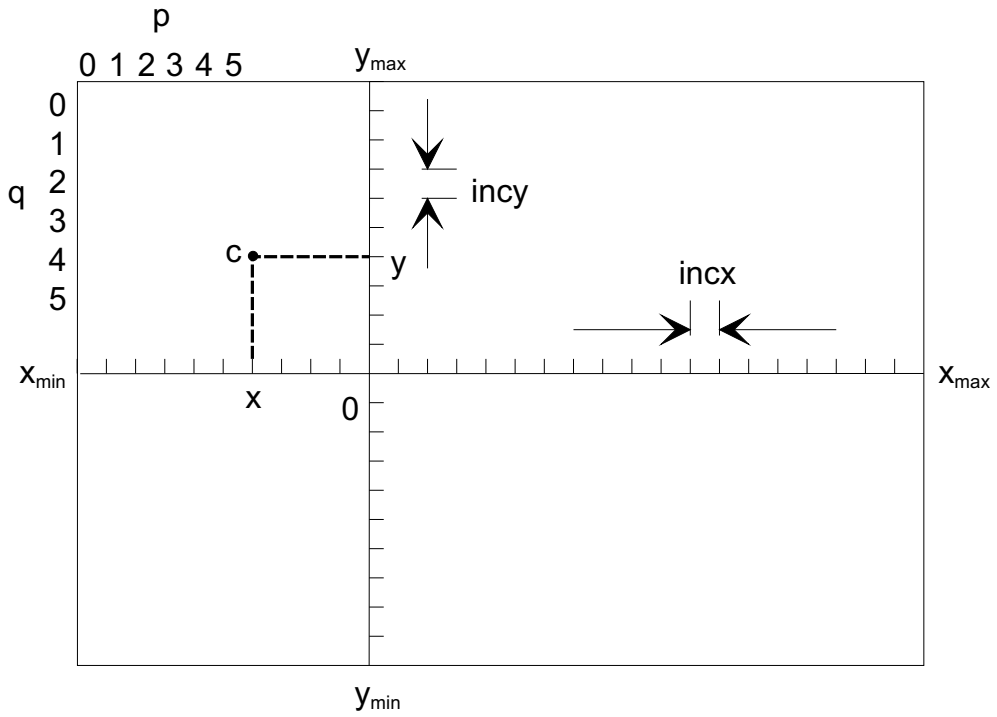
où c et z sont des nombres complexes.

L'ensemble de Mandelbrot, c'est-à-dire la région noire de notre monde de Tor'Bled-Nam, est précisément la région du plan complexe constituée par les points c pour lesquels la suite z_n est bornée. L'image précédente

a été produite par ordinateur. L'ordinateur a fait défiler systématiquement les choix possibles du nombre complexe c, engendre la suite $0, c, c^2 + c \ldots$ et décide, selon un critère approprié, si la suite demeure bornée ou non. Si elle est effectivement bornée, alors l'ordinateur fait en sorte qu'une tache noire apparaisse sur l'écran au point correspondant à c. Si elle n'est pas bornée, il produit une tache blanche.

Le but de l'exercice est d'écrire un programme en C capable de générer l'ensemble de Mandelbrot pour une partie du plan complexe définie par l'utilisateur. Pour chaque nombre complexe c appartenant à cette partie du plan, on détermine si la suite récurrente z_n est bornée ou non. L'ensemble ne sera pas dessiné à l'aide de points noirs ou blancs, mais stockés dans une matrice contenant des 1 ou des 0 selon que la suite converge ou non. Cette matrice sera alors enregistrée dans un fichier pour être éventuellement tracée par un logiciel approprié.

La figure suivante représente les notations utilisées pour définir la portion de plan complexe (autrement dit toutes les valeurs possibles de c) dans laquelle nous allons déterminer l'ensemble de Mandelbrot. On note $c = x + iy$.



1. Écrire un programme qui déclare les variables réelles suivantes :
   $xmin$, $xmax$, $ymin$, $ymax$, $incx$ et $incy$
   puis le tableau bidimentionnel Mandelbrot contenant des entiers, et de dimensions maximales 1024×1024.

2. Demandez à l'utilisateur d'entrer les valeurs de $xmin$, $xmax$, $ymin$, $ymax$, $incx$ et $incy$.
   Calculez alors le nombre de points nx et ny suivant les deux dimensions (attention, nx et ny sont des entiers).

3. Exprimez x en fonction de $xmin$, p et $incx$.
   De la même manière, exprimez y en fonction de $ymin$, q et $incy$.

4. Complétez le programme en y ajoutant deux boucles imbriquées, l'une avec p et l'autre avec q, dans laquelle on calcule la partie réelle x et la partie imaginaire y du nombre complexe c.

5. Écrire la fonction *int checkpoint(float x , float y)* qui teste si la suite récurrente :

$$\begin{cases} z_{n+1} = z_n^2 + c \\ z_0 = 0 \end{cases}$$

est bornée ou non. La fonction renvoie **1** si la suite est bornée, et **0** sinon.
En utilisant une boucle **for**, calculez les 15 premiers termes de la suite z_n. On considère que la suite est bornée si le module carré de z_{15} est supérieur à 10. Le langage C ne disposant pas de type de variable complexe, vous travaillerez avec les parties réelle et imaginaire de z_n en posant :

$$z_n = rez + i\ imz$$

puis en déterminant $module\_carre = rez * rez + imz * imz$

6. Dans les boucles imbriquées précédentes, déterminez si la suite récurrente z_n associée au nombre complexe $c = x + iy$ est bornée ou non, en appelant la fontion *checkpoint(x,y)* et en plaçant la valeur retournée dans la matrice Mandelbrot[p][q].

7. Écrire une fonction qui enregistre la matrice *Mandelbrot* dans un fichier ASCII (caractères «0» et «1») en utilisant le prototype suivant :

void enregistre_Mandelbrot(char fichier[] , int matrice[1024][1024] , int nx , int ny)

fichier	est le nom du fichier dans lequel on veut enregistrer la matrice
matrice	est une copie de la matrice Mandelbrot
nx, ny	sont les dimensions de la matrice

Après avoir ouvert *fichier* en écriture, vous écrirez *matrice[p][q]* dans celui-ci en utilisant la fonction *fprintf(...)* placée dans deux boucles **for** imbriquées.

8. Terminez alors le programme en demandant à l'utilisateur sous quel nom il souhaite enregistrer la matrice, puis appelez la fonction *enregistre_Mandelbrot* avec les paramètres adéquats.

Annexe

---

fopen(nom_du_fichier , "mode")

permet d'ouvrir le fichier *nom_du_fichier*. La fonction renvoie un pointeur vers la mémoire tampon qui sera utilisée pour les opérations de lecture/écriture vers ce fichier. Les différents modes d'ouverture sont les suivants :

r	Ouvre le fichier en lecture. Le pointeur de flux est placé au début du fichier.
r+	Ouvre le fichier en lecture et écriture. Le pointeur de flux est placé au début du fichier.
w	Ouvre le fichier en écriture. Le fichier est créé s'il n'existait pas. S'il existait déjà, sa longueur est ramenée à 0. Le pointeur de flux est placé au début du fichier.
w+	Ouvre le fichier en lecture et écriture. Le fichier est créé s'il n'existait pas. S'il existait deja, sa longueur est ramenée à 0. Le pointeur de flux est placé au début du fichier.
a	Ouvre le fichier en écriture. Le fichier est créé s'il n'existait pas. Le pointeur de flux est placé à la fin du fichier.
a+	Ouvre le fichier en lecture et écriture. Le fichier est créé s'il n'existait pas. Le pointeur de flux est placé à la fin du fichier

exemple :

```
FILE* file_ptr;                        /* déclaration du pointeur */
file_ptr = fopen(''test.dat'',''r''); /* ouverture du fichier */
...
fclose(file_ptr);                      /* fermeture du fichier */
```

fprintf(file_ptr , format , variable(s))

permet d'écrire une chaîne formatée dans le fichier correspondant au pointeur file_ptr.

exemple :

```
fprintf( file_ptr , ''%d'' , x );
```

permet d'écrire la variable entière x dans le fichier *file_ptr*.