

# Nombres complexes et ensemble de Mandelbrot

*Mini-projet de Langage C*

Le but final de ce projet est d'implémenter un petit algorithme qui permet de constituer l'ensemble de Mandelbrot.

## 1 Présentation générale

La *suite de Mandelbrot* est la suite  $(z_n) \in \mathbb{C}^{\mathbb{N}}$  de nombres complexes<sup>1</sup> définie par récurrence ainsi, pour un  $c \in \mathbb{C}$  donné :

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

L'*ensemble de Mandelbrot*<sup>2</sup> est l'ensemble des nombres complexes  $c$  tels que la suite de Mandelbrot soit bornée. Par un tour de passe-passe mathématique, on peut démontrer que s'il existe  $N \in \mathbb{N}$  tel que  $|z_N| > 2$  ( $|z|$  étant le *module* du nombre complexe  $z$ ), alors la suite est divergente. On note  $N(c)$  la *vitesse de divergence* de la suite  $(z_n)$  pour le paramètre  $c$  (on peut considérer que  $N(c) = \infty$  si  $(z_n)$  est bornée pour ce  $c$  là).

On peut alors calculer  $N(c)$  en tout point du plan complexe. En affectant une couleur à  $N(c)$  (noir si  $\infty$  et choisie dans une échelle judicieusement adaptée sinon), on peut donner une représentation graphique de l'ensemble de Mandelbrot.

Comme on le voit sur les images de la figure 1, l'ensemble de Mandelbrot présente une structure fractale riche et étonnante, avec des tourbillons, des îles, et cette figure de la « tique » qui se répète à l'infini.

L'objectif de ce projet est de compléter le visualisateur donné pour qu'il affiche la fractale de Mandelbrot. Le gros du travail (affichage et rendu) a été fait, il ne manque que l'algorithme qui calcule la vitesse de divergence !

## 2 Fourniture et travail demandé

Vous trouverez sur Moodle une archive contenant le projet. Ce projet se compose d'un grand nombre de modules relativement compliqués, auquel vous n'aurez pas à toucher. Les modules/fichiers qui nous intéressent sont les suivants :

- **complexe.h/complexe.c** : module définissant les nombres complexes, à compléter
- **mandelbrot.h/mandelbrot.c** : module spécifiant l'algorithme de calcul de la vitesse de divergence, à compléter
- **test\_complexe.c** : fichier de test pour le module **complexe**, fourni

Le projet se déroule comme suit :

1. Créer le module **complexe** avec la spécification donnée (ci-dessous)
2. Lancer les tests sur le module complexe
3. Compléter le module **mandelbrot** avec l'algorithme spécifié (ci-dessous)
4. Lancer le visualisateur et explorer l'ensemble de Mandelbrot !

1. [https://fr.wikipedia.org/wiki/Nombre\\_complexe](https://fr.wikipedia.org/wiki/Nombre_complexe)  
2. [https://fr.wikipedia.org/wiki/Ensemble\\_de\\_Mandelbrot](https://fr.wikipedia.org/wiki/Ensemble_de_Mandelbrot)

## 2.1 Le module **complexe**

Écrire la spécification (.h) et l’implantation (.c) du module **complexe**. Des morceaux de fichier vous ont été fournis, mais il manque beaucoup de choses !

**ATTENTION :** vous **DEVEZ** respecter les noms et l’ordre des arguments donnés dans le sujet. Sans cela, le programme ne compilera pas !

Voici les étapes à réaliser :

1. Définir le *type utilisateur* **complexe\_t** qui représente un nombre complexe. Utiliser un alias pour que le type qui apparaisse dans le code soit bien **complexe\_t**.
2. Spécifier (= donner contrat et signature) et implanter les fonctions :
  - **reelle**, qui à un nombre complexe associe sa partie réelle
  - **imaginaire**, qui à un nombre complexe associe sa partie imaginaire
3. Spécifier et implanter les procédures :
  - **set\_reelle**, qui modifie la partie réelle du nombre complexe donné avec le nombre réel donné (dans cet ordre)
  - **set\_imaginaire**, qui modifie la partie imaginaire du nombre complexe donné avec le nombre réel donné (dans cet ordre)
  - **init**, qui modifie la partie réelle et la partie imaginaire du nombre complexe donné avec les deux réels donnés (partie réelle puis imaginaire, dans cet ordre)
4. Implanter la procédure **copie** spécifiée dans **complexe.h**, qui modifie le nombre complexe donné de façon à ce que ses composantes soient les même que l’autre nombre passé en paramètre.
5. Implanter les procédures dont les contrats sont données dans **complexe.h** :
  - **conjugue** : calcule le conjugué<sup>3</sup> d’un nombre complexe
  - **ajouter** : réalise l’addition de deux nombres complexes
  - **soustraire** : réalise la soustraction de deux nombres complexes
  - **multiplier** : réalise la multiplication de deux nombre complexes
  - **echelle** : multiplie un nombre complexe par un nombre réel
6. Implanter la procédure **puissance** spécifiée dans le .h. Attention au contrat.
7. Spécifier et implanter les fonctions :
  - **module\_carre**, qui calcule le carré du module<sup>4</sup> du complexe donné en paramètre
  - **module**, qui calcule le module du complexe donné en paramètre (on pourra s’aider de la librairie **math**)
  - **argument**, qui calcule l’argument<sup>5</sup> du complexe donné en paramètre (on pourra s’aider de la librairie **math**) ; attention au contrat de cette fonction

3. <https://fr.wikipedia.org/wiki/Conjugue%C3%A9>

4. [https://fr.wikipedia.org/wiki/Module\\_d'un\\_nombre\\_complexe](https://fr.wikipedia.org/wiki/Module_d'un_nombre_complexe)

5. [https://fr.wikipedia.org/wiki/Argument\\_d'un\\_nombre\\_complexe](https://fr.wikipedia.org/wiki/Argument_d'un_nombre_complexe)

Nous vous fournissons un script qui teste (grossièrement) la conformance de votre `.h` à la spécification demandée (`test_struct`), ainsi qu'une batterie de *tests unitaires* pour tester le module `complexe` (`test_complexe.c`). **Il ne vous est pas autorisé de les modifier.**

Lorsque votre module `complexe` est terminé, vous pouvez le tester en réalisant les étapes suivantes (dans un terminal à l'endroit où se trouve le code) :

```
> make test  
> make runtest
```

(bien sûr, on ne peut faire `make runtest` que si `make test` n'a pas renvoyé d'erreur)

L'exécution de `make runtest` lance les tests et vous donne le résultat (il sauvegarde aussi ces résultats dans un CSV, pour une version synthétique). Si vous avez 0 échecs, vous pouvez passer à l'étape suivante !

## 2.2 Le module `mandelbrot`

Le module `mandelbrot` (composé du header `mandelbrot.h` et du fichier source `mandelbrot.c`) contient l'algorithme de calcul de la vitesse de divergence de la suite de Mandelbrot. Cet algorithme est à implémenter dans la fonction `mandelbrot`, dont le contrat exhaustif et la signature sont données dans le fichier header.

La fonction `mandelbrot` prend en paramètre :

- `z0` le terme initial de la suite de Mandelbrot (a priori  $0 + 0i$  mais la fonction doit pouvoir marcher avec n'importe quelle valeur)
- `c` le terme constant de la suite, qui varie en fonction d'où l'on se trouve dans le plan complexe
- `seuil` le seuil à partir duquel on considère que la suite est divergente
- `maxit` le nombre maximal d'itération à effectuer avant de conclure que la suite est convergente

L'algorithme marche comme suit :

1. On calcule un à un les termes de la suite de Mandelbrot (en suivant la relation de récurrence présentée plus haut)
2. Dès que le module du terme actuel dépasse `seuil`, on s'arrête
3. Dès que le nombre d'itérations effectuées dépasse `maxit`, on s'arrête
4. On retourne le nombre d'itérations effectuées (qui doit donc se trouver entre 0 et `maxit` inclus)

Une fois cet algorithme implémenté, on peut compiler et lancer l'outil :

```
> make  
> ./mandelbrot
```

L'outil ouvre une fenêtre et réalise le rendu de la fractale de Mandelbrot en utilisant votre algorithme ! Vous devez voir apparaître la fameuse « tique » de Mandelbrot.

On peut faire un click gauche à un endroit pour zoomer dessus, ou un click droit pour dézoomer. On peut aussi déplacer l'image vers le haut, le bas, la gauche ou la droite en utilisant les flèches directionnelles du clavier. On peut faire varier la luminosité de l'image avec P (plus) et M (moins), ce qui est pratique quand on arrive dans des zones « très lumineuses ».

Enfin, il suffit de faire Q ou Échap (ou de cliquer sur la croix) pour quitter la fenêtre.

À noter que la fenêtre traque la souris et affiche les coordonnées complexes de sa position dans l'image. Les mensurations du repère sont par ailleurs affichées en bas à droite de la fenêtre.

À noter qu'on peut configurer l'outil au lancement en utilisant divers arguments. La liste et la description des arguments acceptés par l'outil est accessible en faisant `./mandelbrot -help`.

En particulier, dans un premier temps, il peut être utile de réduire la taille de la fenêtre (pour qu'il y ait moins à calculer et donc obtenir une image plus vite, bien que moins détaillée). On peut choisir une fenêtre en 600 par 800 avec les options `-h 600 -w 800`.

On peut aussi réduire le nombre d'itérations maximal pour l'algorithme (plus il est bas et plus l'algorithme termine rapidement ; par contre, il perd aussi beaucoup en précision...). Dans un premier temps par exemple, on peut se limiter à 1000 itérations, avec `-i 1000`.

Cela donne, pour cet exemple de configuration :

```
> ./mandelbrot -h 600 -w 800 -i 1000
```

### 3 Modalités de rendu et d'évaluation

Le rendu se fait **sur Moodle**, sur la page de la matière à l'endroit indiqué. Il ne doit y avoir **qu'un seul rendu par groupe**.

Vous devrez déposer sur Moodle une **archive .tar.gz** que l'on obtient en utilisant le script fourni, `creer_rendu.sh`. Pour cela, vous pouvez simplement faire : `./creer_rendu.sh` ou `make rendu`. La commande crée un fichier `rendu.tar.gz`, que vous n'avez plus qu'à déposer sur Moodle.

Pour votre information, l'archive contiendra les fichiers `complexe.h`, `complexe.c`, `mandelbrot.c` et `IDENT.txt` (fichier d'identification généré par le script) *et rien d'autre*. Il ne vous est pas permis de rendre des fichiers additionnels ou d'autres fichiers que vous auriez modifié.

Lors de la correction, les fichiers sont placés dans un environnement contrôlé (le même que celui qui vous est fourni, en fait) et testé dans ces conditions.

**Nous attendons de vous que :**

1. **vous respectiez scrupuleusement les consignes énoncées dans cette partie (utilisation du script, notamment)**
2. **les fichiers rendus compilent *sans erreur* dans l'environnement (celui des machines de l'ENSEEIHT)**

En cas de non respect de l'une de ces deux règles, **la note sera automatiquement mise à 0**.

Par ailleurs, notez que **l'utilisation de modèles de langage (ChatGPT, Copilot, etc.) est strictement interdite, de même que la recopie depuis un autre groupe**. Tout triche entraînera des sanctions.

Les rendus sont compilés, testés et examinés par votre intervenant de TP. Ils sont évalués selon 3 critères principaux :

1. Fonctionnalité du code
2. Respect des consignes
3. Qualité du code

Pour avoir le maximum de points, il faut notamment que la compilation n'entraîne aucun warning (les options `-Wall -Wextra -pedantic` sont utilisées), et que *tous les tests unitaires* passent. Les contrats doivent être complets et exhaustifs (et respecter la spécification présentée dans le sujet du projet), et le code doit respecter les bonnes pratiques de la programmation que vous devez maîtriser (indentation correcte, nom de variable qui a du sens, utilisation des bonnes structures, etc.).

**Pour toute question, s'adresser à votre encadrant de TP. Des informations complémentaires (réponses à des questions courantes, dates de rendu) peuvent se trouver sur la page Moodle de la matière, à consulter de temps à autres pour ne rien rater !**

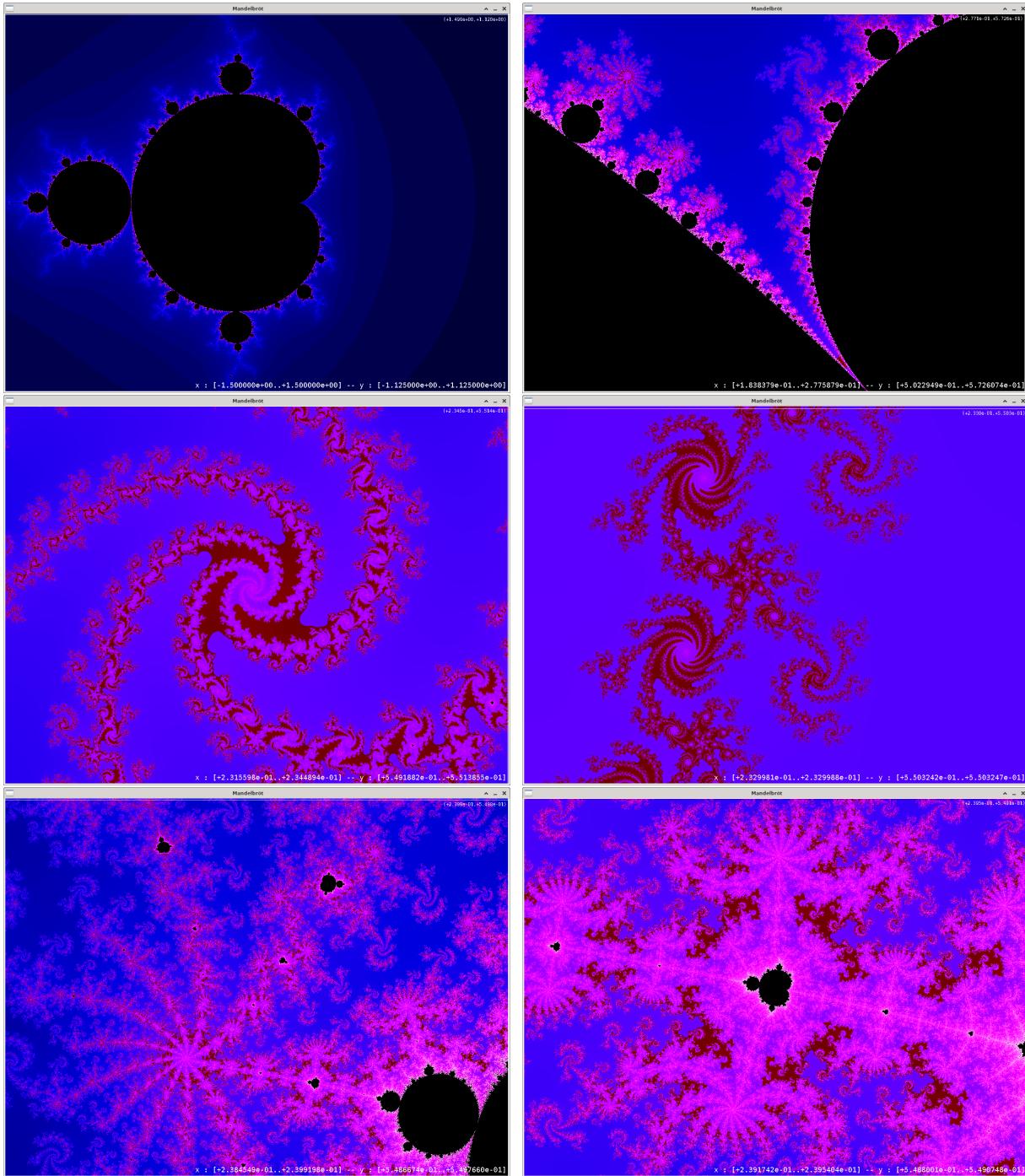


FIGURE 1 – Exemple de zones de la fractale de Mandelbrot, obtenues avec une implantation de l'outil développé dans le projet