Ecole Centrale de Marseille - 2022-2023

S8 SISN - U.E. CRI

Introduction à la programmation en langage C

Enoncé de TP

Email : *nicolas.bertaux@centrale-marseille.fr*

Dans ce module, vous aurez 2 séances de 4 heures de TP.

A la fin de chacune des 2 séances de TP, vous devrez nous envoyer par email les codes que vous avez écrits, et qui serviront de base à votre évaluation.

Objectif : Durant ces 2 séances, l'objectif est de créer en langage C un programme permettant de visualiser les évolutions d'un automate cellulaire, et plus précisément l'automate cellulaire créé par Conway et connu sous le nom de jeu de la vie :
https://fr.wikipedia.org/wiki/Jeu_de_la_vie

Les règles du jeu de la vie sont les suivantes :

- On dispose d'un tableau 2D de dimension $N \times M$, chacune des cases comportant une cellule qui peut être soit morte (valeur 0 dans le tableau) soit vivante (valeur 1).
- A partir de la configuration de ce tableau à l'étape numéro i, on peut déterminer la configuration du tableau à l'étape $i+1$.

Plus précisément, à chaque itération, l'évolution d'une cellule est entièrement déterminée par l'état de ses huit voisines à l'itération précédente, selon les 2 règles suivantes :

- Naissance d'une cellule : une cellule morte à l'itération i devient vivante à l'itération $i+1$ si et seulement si elle possédait exactement 3 voisines vivantes à l'itération i.
- Mort d'une cellule : une cellule vivante à l'itération i reste vivante à l'itération $i+1$ si et seulement si elle possédait 2 ou 3 voisines vivantes à l'itération i (sinon, elle meurt).

Vous devrez concevoir un programme permettant de visualiser à partir d'une initialisation aléatoire, la configuration de votre automate cellulaire au bout de K itérations.

Consignes à respecter :

- Vous partirez d'un fichier vierge et vous pourrez mettre votre fonction `main` ainsi que l'ensemble de vos fonctions dans ce même fichier.
- **Vous devrez compiler et tester votre programme vraiment très fréquemment, quasiment chaque fois que vous ajoutez une ligne.** Pour un premier programme réalisé en C, ce sujet est assez ambitieux, mais si vous respectez cette consigne, cela vous permettra de passer vraiment très peu de temps à débugguer votre programme. Si vous ne le faites pas, cela risque d'être difficile.
- Votre programme ne doit pas être constitué d'une longue fonction `main` comportant tout votre code, mais au contraire d'une fonction `main` courte faisant appel à différentes fonctions que vous aurez créées.
- Votre code devra être lisible, commenté, et chaque fonction devra être accompagnée d'une courte description (une ligne ou deux).

Pour vous aider dans la réalisation de votre programme, voici comment vous pourrez structurer votre `main` :

1. Lecture des paramètres fournis par l'utilisateur : pour cela, vous créerez une fonction permettant de récupérer ces paramètres sur la ligne de commande (notamment les dimensions N et M du jeu de la vie, ainsi que le nombre d'itérations K souhaitées). Vous prévoirez également que lorsque l'utilisateur ne fournit pas le bon nombre de paramètres, le programme affiche l'aide à l'écran.

2. Allocation de la mémoire : dans la mesure où vous devrez manipuler des matrices de taille M par N, vous devrez créer un fonction d'allocation dynamique de tableaux 2D.

3. Initialisation du jeu de la vie : là encore, vous devrez faire appel à une (ou plusieurs) fonction dédiée que vous aurez créé.

4. Boucle sur K itérations, où à chaque itération, vous calculerez l'itération suivante (en faisant appel à une fonction, elle-même pouvant être composée de sous-fonctions).

5. Affichage du résultat obtenu sur le terminal, en faisant appel à une fonction d'affichage. Pour une meilleure visibilité, vous afficherez les cellules vivantes avec des lettres 'o' et les cellules mortes avec des espaces ' '. Vous pourrez également délimiter les bords de votre tableau sur le terminal a l'aide des caractères '−' et '|'.

6. Libération de la mémoire : vous penserez bien à désallouer les différents tableaux que vous aurez alloués. Ici aussi, vous devrez avoir créé une fonction de désallocation pour les tableaux 2D. Vous vérifierez que vous avez bien utilisé autant de fois la fonction `free` que la fonction `malloc`.

Afin de respecter la consigne vous demandant de compiler/tester quasiment à chaque ligne de code ajoutée, il est clair que vous ne devez pas coder votre programme dans n'importe quel ordre. En particulier, l'erreur fréquente consiste à commencer par coder l'étape (4) de votre `main` (i.e. codage de l'algorithme d'évolution du jeu de la vie) : il ne vous sera alors pas possible de tester votre programme, ni même simplement de le compiler, dans la mesure où vous ne saurez pas encore ni allouer de tableau, ni initialiser ou afficher une carte du jeu de la vie.
Nous vous conseillons donc de réaliser les différentes étapes dans l'ordre suivant : étape **1** (lecture des paramètres), étape **2** (allocation), étape **6** (désallocation), étape **5** (visualisation), étape **3** (initialisation) et enfin étape **4** (calcul des itérations).

Pour un meilleur rendu, vous pourrez ajouter une visualisation après l'initialisation, ainsi qu'après le calcul de chacune des itérations. Pour cela, vous pourriez avoir besoin de la fonction suivante :

```
usleep(1000);
```

qui impose au programme de faire une pause (mesurée en microsecondes), et qui vous permettra d'attendre ainsi entre chaque affichage. Il faudra également ajouter en entête de votre fichier

```
#include <unistd.h> /* Pour la fonction usleep */
```

Vous pourrez également utiliser la commande

```
system("clear");
```

qui lance la fonction `clear` sur le terminal, vous permettant ainsi d'effacer la fenêtre du terminal entre deux affichages.

Génération de nombres aléatoires : Si vous avez besoin de générer des nombres aléatoires (par exemple pour initialiser votre jeu de la vie), vous devez tout d'abord penser à initialiser la graine aléatoire, en insérant dans votre `main` (avant d'avoir effectué des tirages aléatoires), le code suivant :

```
srand(clock()); /* initialisation de random sur l'horloge */
```

Vous devrez également ajouter dans l'entête de votre fichier :

```
#include <time.h> /* pour la fonction clock */
```

Si vous voulez effectuer un tirage aléatoire uniforme entre 0 et 1 (que ce soit dans le `main` ou à l'intérieur de n'importe quelle fonction), il vous suffira d'écrire

```
double r;
r = (double)rand()/RAND_MAX;
```