Examen – 8 janvier 2019
ING1 – Programmation C
Génie Informatique

Modalités

— Durée : 3 heures

— Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.

— Aucun document papier n'est autorisé.

— Aucun déplacement n'est autorisé.

— Aucun échange, de quelle que nature que ce soit, n'est possible.

— La présence d'un Makefile et des commentaires doxygens sont un plus, et seront appréciés.

— La lisibilité du code et les commentaires seront pris en compte dans la notation, de même que la séparation des différentes procédures/fonctions dans des fichiers d'en-tête (.c .h).

— Tout code qui ne compile pas entrainera une **pénalité de 50%. Si une partie de votre code ne compile pas, commentez-le.** Ceci implique évidemment que vous compiliez au fur et à mesure!

— Chaque warning de compilation entrainera une **pénalité de 15%**.

— Tout ce qui sera dans le dossier **rendu** correspondra à votre rendu, et sera récupéré à la fin de l'examen. Je vous conseille vivement de travailler directement dans ce dossier.

1 Introduction

Le « jeu de la vie »(CONWAY, 1970) est un modèle simpliste de l'évolution d'une population de cellules qui naissent, vivent et meurent en fonction de leur voisinage. Les cellules sont disposées sur une grille (matrice carrée), dont chaque case peut contenir une cellule ou être vide à une étape donnée. La figure 1 ci-dessous présente un exemple de grille de dimension 5 × 5.

	0	1	2	3	4
0	B			D	O
1	F			H	C
2					
3					
4	E			I	A

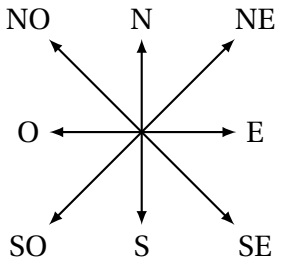

FIGURE 1 – Grille torique de dimension 5 × 5

La grille est dite torique, c'est-à-dire que chaque ligne (et chaque colonne) reboucle sur elle-même. Chaque case dispose donc de 8 cases voisines. Par exemple dans le cas de la case de coordonnées (0, 4) contenant la lettre O sur la figure 1, on a les 8 cases voisines suivantes :

— la case A située au nord en (4, 4) ;

— la case E située au nord-est en (4, 0) ;

— la case B située à l'est en (0, 0) ;

— la case F située au sud-est en (1, 0) ;

— la case C située au sud en (1, 4) ;

— la case H située au sud-ouest en (1, 3) ;

— la case D située à l'ouest en (0, 3) ;

— la case I située au nord-ouest en (4, 3).

L'évolution de la grille se fait au tour par tour en appliquant un ensemble de règles locales à chaque case. Ces règles prennent en compte l'état et le voisinage de la case et sont appliquées simultanément sur toutes les cases de la grille.

2 Simulation d'une prairie

Une des évolutions du jeu de la vie est de simuler le développement d'herbes dans une prairie. Le programme doit faire évoluer chaque herbe dans une prairie symbolisée par une grille carrée

et torique. Une herbe est caractérisée par :

1. une énergie qui augmente d'une constante +`ENERGIE = +4` à chaque tour de la simulation ;
2. un âge qui augmente d'une unité à chaque tour de la simulation ;
3. une position sur la prairie.

Chaque herbe dont l'âge est égal à la constante `AGE_MORT = 5` meurt immédiatement. Chaque herbe dont l'âge est strictement inférieur à la constante `AGE_MORT` et dont l'énergie est supérieure ou égale à la constante `ENERGIE_REPRODUCTION = 10` tente de se reproduire sur toutes les cases voisines (dans les 8 directions) non encore occupées (la reproduction est impossible si toutes les cases voisines sont déjà occupées). Cette reproduction (si elle est possible) lui côute `ENERGIE_REPRODUCTION` points d'énergie. Chaque nouvelle herbe crée dans la prairie l'est avec une énergie initiale égale à la constante `ENERGIE_INITIALE = 1` et avec un âge nul.

Nous allons donc, pour cet examen, réaliser la simulation d'une prairie.

3 Implémentation

(1) Définir les constantes symboliques `AGE_MORT`, `ENERGIE`, `ENERGIE_REPRODUCTION` et `ENERGIE_INITIALE`.
Définir la structure `sHerbe`. □

(2) Écrire la fonction `allouerGrille` qui, pour une dimension donnée, alloue l'espace nécessaire pour stocker la prairie. □

(3) Écrire la procédure `creationAleatoire` qui, pour une grille d'une dimension donnée, crée 10% de l'herbe répartie de façon aléatoire sur la grille. □

(4) Écrire la procédure `afficherGrille` qui prend en paramètres une grille et sa dimension et l'affiche sur la sortie standard en symbolisant les cases vides par un point et l'herbe par la lettre 'h'. □

(5) Écrire la fonction `reproduireAutour` qui prend en paramètres une grille, sa dimension et la position de l'herbe, qui tente de se reproduire sur chacun des 8 voisins. La fonction renvoit le nombre de fois où elle a pu se reproduire. □

(6) Écrire la procédure `prochaineGeneration` qui calcule la nouvelle génération de la grille donnée. □

(7) Écrire la procédure `simulerPrairie` qui prend en paramètre une grille et sa dimension et qui effectue le travail suivant :

- affiche la grille ;
- tant que l'utilisateur s'obstine à ne pas répondre `oui` ou `non` à la question `nouvelle generation ? (oui/ non)`, `generation` est réaffichée et la question reposée ;
- si la réponse est `oui`, la nouvelle génération est calculée, affichée puis la question précédente est reposée à l'utilisateur ;
- si la réponse est `non`, la fonction termine son éxécution. □

(8) Écrire la procédure `sauvegarderPrairie` qui permet de sauvegarder dans le fichier *prairie.txt* l'état de la grille. On stockera dans le fichier la dimension de la grille, et la valeur de l'age et de l'énergie pour chaque herbe de la prairie. □

(9) Écrire la fonction `creationDepuisFichier` qui permet d'initialiser une grille à partir des valeurs stockées dans un fichier. Le nom du fichier sera donné en paramètre de la fonction, ainsi que la dimension de la grille, et la fonction retournera la grille créée et initialisée. □

(10) Le programme principal pourra être éxécuté de deux façons différentes :

- Aucun argument n'est donné au programme, donc la taille de la prairie est demandée à l'utilisateur et la prairie est initialisée aléatoirement.
- Un argument est donné au programme. Cet argument correspond au nom du fichier dans lequel est stocké une prairie. Elle sera donc créée et initialisée en fonction des valeurs données dans le fichier.

Avant de quitter le programme, la grille courante sera sauvegardée. □

Remarque : Un exemple de fichier pour sauvegarder/créer une prairie vous est donné dans le fichier *exemple-prairie.txt*.