ING1 : RATTRAPAGE DE PROGRAMMATION C
ING1 – Programmation C
23/02/2012

1 Préambule

Cet examen dure 3h00. Le rendu de l'examen s'effectuera par le script `rendu_exam_ing1`, qui prend en argument le dossier à rendre. Pendant cette épreuve, aucun document papier n'est autorisé. Vous ne devrez être connecté qu'à un seul ordinateur. Une multiple connexion sera considéré comme de la triche, et entrainera un 0.

Les pénalités sont :
- **-50% de la note s'il y a une erreur de compilation**
- **-20% de la note par warning de compilation**
- **jusqu'à -2 s'il y a des erreurs/warnings de doxygen, ou si les règles de programmation de sont pas respectées**

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). Il vous appartient, pour chaque fonction demandée, de respecter le prototype. Vous êtes libres d'ajouter toutes les fonctions et procédures qui vous paraissent nécessaires. La gestion de la mémoire (allocation, libération) devra être géré.
Le barème est donné à titre indicatif.

2 Le jeu de la vie

2.1 Introduction

Le "*jeu de la vie*" (CONWAY, 1970) modélise l'évolution d'un ensemble de cellules qui naissent, vivent et meurent en fonction de leur voisinage. L'espace consiste en une matrice carrée ("*grille*") où chaque cellule (chaque case) est, à une étape donnée, une cellule vivante ou une cellule morte. La figure 1 présente un exemple de grille de dimension 5×5.

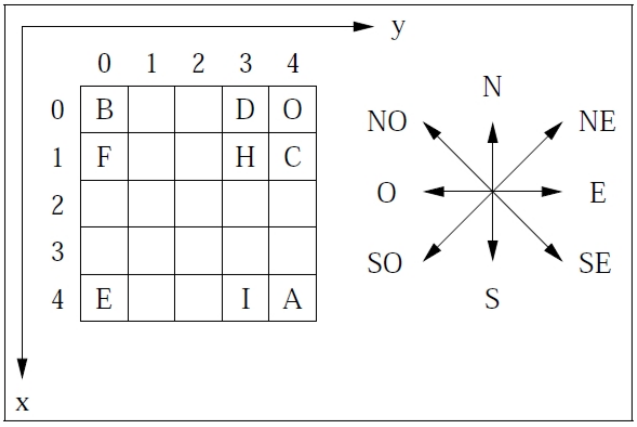


FIGURE 1 – Grille torique de dimension 5 × 5

Cette grille est gérée à la manière d'une grille torique, c'est à dire une grille rebouclée sur elle-même (sphère). De fait, chaque case dispose d'exactement 8 cases voisines. Ainsi, dans le cas de la case O de coordonnées ($x = 0$, $y = 4$), nous avons :

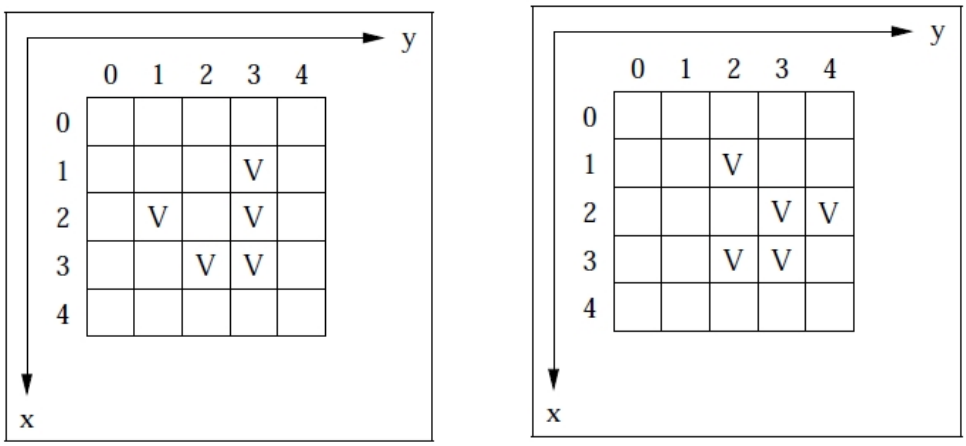
- la case A située au Nord (direction N) de O a pour coordonnées ($x = 4$, $y = 4$);
- la case B située à l'Est (direction E) de O a pour coordonnées ($x = 0$, $y = 0$);
- la case C située au Sud (direction S) de O a pour coordonnées ($x = 1$, $y = 4$);
- la case D située à l'Ouest (direction O) de O a pour coordonnées ($x = 0$, $y = 3$);
- la case E située au Nord-Est (direction NE) de O a pour coordonnées ($x = 4$, $y = 0$);
- la case F située au Sud-Est (direction SE) de O a pour coordonnées ($x = 1$, $y = 0$);
- la case H située au Sud-Ouest (direction SO) de O a pour coordonnées ($x = 1$, $y = 3$);
- la case I située au Nord-Ouest (direction NO) de O a pour coordonnées ($x = 4$, $y = 3$);

Le jeu consiste, à partir d'une grille initiale contenant des cellules vivantes et des cellules mortes, à calculer la prochaine grille en appliquant les règles suivantes (les cases de la nouvelle grille sont calculées simultanément à partir de celles de la grille précédente):

1. *règle régissant la survie* : chaque cellule vivante disposant de 2 ou 3 voisines vivantes continue de vivre;
2. *règle régissant la mort* : chaque cellule vivante disposant de 4 voisines vivantes ou plus meure de surpopulation. Chaque cellule vivante entourée de 0 ou 1 cellule vivante meure d'isolement;
3. *règle régissant la naissance* : chaque cellule morte disposant d'exactement 3 voisines vivantes devient vivante.

La figure 2a présente une grille initiale dans laquelle les cellules vivantes sont représentées par la lettre `V` (les cellules mortes ne sont pas représentées). La figure 2b présente la grille suivante

obtenue en appliquant les trois règles précédentes.



(a) étape $t = 0$ (b) étape $t = 1$

FIGURE 2 – Évolution de la Grille

2.2 Structure des données

Nous allons représenter la grille sous la forme d'une structure contenant un tableau à deux dimensions ainsi qu'une taille. Afin d'assurer l'existence des 8 voisins, la grille aura une taille minimum de 3. Pour faciliter le problème, la grille sera toujours une matrice carré.

type Grille = **structure**
  matrice: Tableau de entier
  taille: entier
fin type

Les cases de la matrice peuvent prendre deux valeurs possibles :

- la valeur 0 désignant une cellule morte;
- la valeur 1 désignant une cellule vivante.

Ces deux valeurs seront symbolisées par les constantes globales `MORT = 0` et `VIE = 1`.

(1) **Structure - 1 point**
Créer la structure `Grille`. Étant donné que la taille de la grille n'est pas connu à l'avance, la matrice sera dynamique. Définir les deux constantes globales. □

2.3 Fonctionnalités

(2) **Constructeur - 2 points**
Créer la fonction `creerGrille()` qui demande à l'utilisateur la taille de la grille, crée et initialise la grille aléatoirement et la retourne. □

(3) **Affichage - 2 points**
Créer la procédure `afficherGrille(grille)` qui affiche la grille. Les cellules mortes seront représentées par le '.' et les cellules vivantes par la lettre 'V'. □

(4) **Valeur des voisins - 3 points**

Afin de faciliter l'implémentation des règles du jeu, créer les 8 fonctions suivantes qui permettent d'accéder à la valeur des cellules voisines.

- `valeurNordGrille(grille, x, y)` qui retourne la valeur de la case de `grille` situé au Nord de la case (`x`,`y`);
- `valeurEstGrille(grille, x, y)` qui retourne la valeur de la case de `grille` situé à l'Est de la case (`x`,`y`);
- `valeurSudGrille(grille, x, y)` qui retourne la valeur de la case de `grille` situé au Sud de la case (`x`,`y`);
- `valeurOuestGrille(grille, x, y)` qui retourne la valeur de la case de `grille` situé à l'Ouest de la case (`x`,`y`);
- `valeurNordEstGrille(grille, x, y)` qui retourne la valeur de la case de `grille` situé au Nord-Est de la case (`x`,`y`);
- `valeurSudEstGrille(grille, x, y)` qui retourne la valeur de la case de `grille` situé au Sud-Est de la case (`x`,`y`);
- `valeurSudOuestGrille(grille, x, y)` qui retourne la valeur de la case de `grille` situé au Sud-Ouest de la case (`x`,`y`);
- `valeurNordOuestGrille(grille, x, y)` qui retourne la valeur de la case de `grille` situé au Nord-Ouest de la case (`x`,`y`). □

(5) **Nouvelle valeur d'une cellule - 3 points**

Créer la fonction `calculerNouvelleValeurCellule(grille,x,y)` qui prend en paramètres une grille et des coordonnées et qui retourne la nouvelle valeur de la case (`x`,`y`) de `grille` calculée en fonction des règles régissant les cellules. □

(6) **Nouvelle valeur d'une grille - 2 points**

Créer la procédure `calculerNouvelleGrille(grille)` qui prend en paramètres une grille et qui crée la grille de l'itération suivante, avant de la retourner. □

7 **Sauvegarde - 2 points**

Créer la procédure `sauvegarderGrille(grille)`, qui écrit la grille dans le fichier "*grilleInitiale*". La structure du fichier sera :

– La première ligne correspond à la taille de la grille
– La grille est affichée sur les autres lignes, avec les valeurs 0 et 1 pour l'état des cellules.

Par exemple :

```
5
1 0 1 1 0
0 0 1 0 0
0 1 1 1 0
0 0 1 1 1
1 0 0 0 1
```

□

8 **Tour de simulation - 3 points**

Créer une méthode `run(grille)` qui prend en paramètre une grille, et qui effectue le travail suivant :

– demander à l'utilisateur s'il désire poursuivre la simulation. La question est reposée tant que l'utilisateur n'entre pas le caractère `o` ou le caractère `n`. Pour vous permettre de récupérer correctement les caractères vous pouvez au choix, utiliser la fonction `scanf` ou `getchar`;
– si la réponse est `o`, afficher la grille courante, calculer la suivante et recommencer;
– si la réponse est `n`, afficher le message – `fin de simulation` – et sortir.

□

9 **Programme - 2 points**

Créer un programme qui crée une nouvelle grille, sauvegarde son état et lance la simulation.

□