LIFAP1 – Séquence 5

Contrôle Continu – TP9 - durée 1h30 mn

Jeudi 15 décembre 2016

Nom :	N° d'étudiant :
Prénom :	

Consignes

Vous devrez compiler et tester votre programme. Vous donnerez votre nom au fichier source. Dans votre fichier, vous mettrez en commentaire vos nom et prénom ainsi que votre numéro d'étudiant.
La note tiendra compte du respect des consignes, de la qualité de la présentation et de la lisibilité du code, des algorithmes, et du bon fonctionnement du programme.

Une fois le programme terminé et testé (ou à la fin du temps imparti), vous devrez déposer le fichier source (.cpp) via **TOMUSS** (en cliquant sur "déposer" dans la case `Depot_TP9` de l'UE LIFAP1).
Vous devrez rendre le sujet à la fin de l'épreuve avec la réponse à la question 1.

Objectif

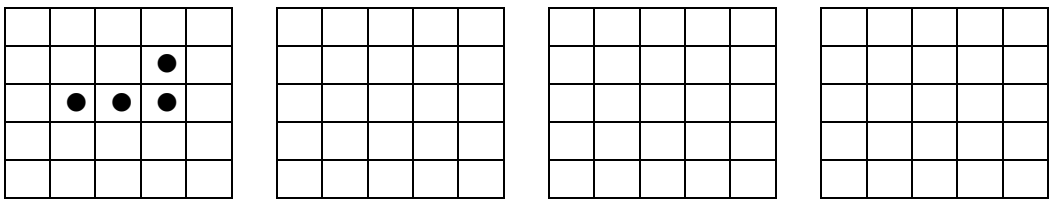
Nous allons écrire un programme permettant la simulation du jeu de la vie. Le jeu de la vie est une grille 2D dans laquelle chaque case peut prendre l'un des deux états « vivant » ou « mort » et est entourée de huit cases susceptibles d'accueillir d'autres cellules.

Les règles d'évolution entre le temps t et le temps t+1 sont les suivantes.

- **La survie** : chaque cellule ayant **deux ou trois** cellules adjacentes vivantes survit jusqu'à la génération suivante.
- **La mort** : chaque cellule ayant **quatre** cellules adjacentes vivantes **ou plus** disparaît, ou meurt, par surpopulation / étouffement. Chaque cellule n'ayant **qu'une ou aucune** cellule adjacente meurt d'isolement.
- **La naissance** : chaque emplacement adjacent ayant **exactement trois** cellules, fait naître une nouvelle cellule pour la génération suivante.

Exercice préliminaire

1. En appliquant les règles énoncées précédemment, prévoyez l'évolution du système suivant.



temps t *temps t+1* *temps t+2* *temps t+3*

Implémentation

2. Définir deux constantes `MAX_X` et `MAX_Y` fixant les dimensions maximales de la grille de jeu ayant chacune la valeur 20.

3. Définir la structure `grille_cell` comportant les champs suivants :
   - `tx` et `ty` les dimensions réelles du jeu,
   - `nb_viv` le nombre de cellules vivantes dans la grille,
   - `nb_mort` le nombre de cellules mortes,
   - et `etat` une grille 2D d'entiers représentant l'état de chaque cellule dans le jeu. L'état sera 0 pour une cellule morte et 1 pour une cellule vivante.

4. Ecrire une fonction `init_grille` permettant de demander à l'utilisateur la taille réelle de la grille (`tx` et `ty`), le nombre de cellules vivantes (`nb_viv`) au départ (et donc de déduire le nombre de cellules mortes `nb_mort`) et d'initialiser toute la grille `etat` avec des cellules mortes (valeur 0).

5. Ecrire une procédure `place_cell_vivantes` qui permet de placer aléatoirement les `nb_viv` cellules sur la grille 2D. Attention, on prendra garde de ne pas mettre plusieurs cellules vivantes dans la même case !

   Rappel : la fonction `rand()` retourne un entier compris entre 0 et une constante RANDMAX. Elle nécessite l'inclusion des bibliothèques `stdlib.h` et `time.h`.

6. Ecrire une procédure `affiche_grille` qui affiche le contenu de la grille d'états.

7. Ecrire la **fonction** `etat_suivant` qui à partir d'une configuration donnée au temps t calcule l'état suivant au temps t+1 en fonction des règles d'évolution énoncées en préambule et retourne le nombre de changements effectués. Pour chacune des cases, on comptera le nombre de voisines vivantes au temps t. En fonction de cette valeur, la cellule restera vivante, naitra, ou mourra sur la grille représentant le temps t+1. L'utilisation de 2 grilles distinctes (à t et à t+1) sera indispensable.

   Remarque : attention à la gestion des cellules du contour !

8. Ecrire le programme principal qui, après avoir initialisé la grille de jeu, placé les cellules vivantes aléatoirement, et affiché la grille dans son état initial, affiche les états successifs de la grille de jeu tant que le nombre de changements est strictement positif et que l'utilisateur souhaite poursuivre le programme (la question lui sera posée à chaque itération).