Examen

Langage Procédural : C

12 mars 2019

La compréhension du sujet fait partie de l'examen. Il ne sera donc répondu à aucune question. Le barème prend en compte l'argumentation fournie dans les commentaires associés aux réponses.

Tout document et fichier source personnel autorisés, communications électroniques interdites. Les librairies standard du C sont autorisées.

Durée 1h30 individuel ; barème approximatif

Pour chaque exercice, vous devez créer une fonction `main` indépendante et fournir vos sources et script de compilation (ou Makefile). Vous posterez l'ensemble de ces sources sous la forme d'une seule archive sur *madoc.univ-nantes.fr* dans l'emplacement prévu dans le cours *Info3 - Langage procédural - C*

Veillez à respecter la séparation de la déclaration du prototype et du corps de vos fonctions.

1 Le jeu de la vie (8 points)

On dispose d'un échiquier, dans lequel les cases peuvent prendre les valeurs 0 ou 1. À chaque étape, on calcule la nouvelle valeur de chacune des cases en fonction des valeurs des cases voisines. Les cases voisines d'une case étant celles se trouvant de part et d'autre de cette case sur la même ligne ou sur la même colonne (pas en diagonale).

La valeur d'une case passe à 0 si elle possède 0 ou 4 voisins à 1 (elle meurt isolée ou étouffée). La valeur d'une case passe à 1 si elle possède 2 ou 3 voisins à 1. Elle ne change pas dans les autres cas.

Pour éviter les cas particuliers des cases se trouvant en périphérie de l'échiquier, on considérera que ce dernier est stocké dans un tableau comprenant une bordure remplie de zéros (voir figure 1).

0	*0*	*0*	*0*	*0*	*0*
0	1	0	1	0	*0*
0	1	1	0	1	*0*
0	1	1	0	0	*0*
0	1	0	0	1	*0*
0	*0*	*0*	*0*	*0*	*0*

FIGURE 1 – Dans l'échiquier représenté ici (avec sa bordure), la case à l'intersection de la $2^{\text{ème}}$ ligne et de la $3^{\text{ème}}$ colonne à 3 voisins à 1.

QUESTION 1. (1 point) Écrire le programme principal déclarant un tableau `M` carré à 2 dimensions (la taille est une constante du programme) puis appelle une fonction `ConfigInit` prenant en paramètre le tableau, demande à l'utilisateur de saisir un nombre d'étapes du jeu de la vie (stockée dans une variable `NbEtapes`) puis effectue une boucle de `NbEtapes` en appelant les fonctions `ChangeEtat` et `Imprime` prenant toutes deux le tableau `M` en paramètre.

QUESTION 2. ($\frac{1}{2}$ point) Écrire la fonction `Imprime` qui prend en paramètre un tableau stockant un échiquier et affiche ce dernier à l'écran.

QUESTION 3. (1 point) On souhaite pouvoir calculer pour une case de l'échiquier le nombre de voisins à 1 afin de tester les différents cas. Écrivez la fonction `nbVoisins` permettant d'obtenir ce nombre. Les paramètres sont à votre discrétion.

QUESTION 4. (1 point) Écrire une fonction `BinRandom` qui retourne au hasard la valeur 0 ou la valeur 1.

QUESTION 5. (1 point) On souhaite initialiser l'échiquier avec une valeur au hasard pour chacune des cases. Écrivez la fonction `ConfigInit` qui permet d'initialiser le tableau, c'est-à-dire non seulement initialiser l'échiquier mais aussi la bordure.

QUESTION 6. ($\frac{1}{2}$ point) En utilisant les fonctions et procédures précédentes, écrivez la fonction `EstVivante` qui retourne la valeur que devra prendre une case à l'étape suivante.

QUESTION 7. (1 point) Complétez la fonction `ChangeEtat` qui permet de faire évoluer l'ensemble de l'échiquier d'une étape vers une autre.

QUESTION 8. (1 point) Créez une fonction `ImprimeDifférence` qui permet d'afficher la différence entre deux états du jeu de la vie donnée en paramètre.

QUESTION 9. (1 point) Modifiez votre programme afin de pouvoir utiliser la fonction `ImprimeDifférence` à la suite de la fonction `Imprime` dans le programme principal.

2 Arbre généalogique (12 points)

On cherche à représenter un arbre généalogique capable de prendre en compte les recompositions familiales. L'ensemble des structures de données nécessaires est représenté, dans la figure 2.

Dans toutes les questions de cet exercice, vous devez établir la liste des paramètres des fonctions, sauf s'ils sont explicitement mentionnés.

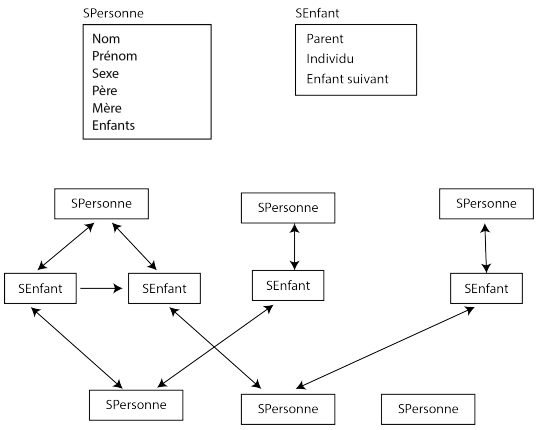

FIGURE 2 – Structures de données permettant de gérer une généalogie.

QUESTION 1. (2 points) Établissez les structures de données `SPersonne` et `SEnfant`.

QUESTION 2. (1 point) Implémentez une fonction `AjoutPersonne` qui créé une personne sans parent ni enfant, ni lien de fratrie. Cette fonction renverra l'adresse mémoire de la personne crée.

QUESTION 3. (1 point) Implémentez une fonction `AfficherPersonne` qui permet d'afficher les détails (Nom, Prénom, Sexe) d'une personne passée en paramètre.

QUESTION 4. (2 points) Implémentez une fonction `AttacherPersonne` qui permet d'ajouter un enfant à un père et une mère. Cette fonction doit de plus établir les liens de fraternité entre enfants d'un même parent.

QUESTION 5. (1 point) Implémentez une fonction `AfficherEnfants` qui permet d'afficher les enfants d'une personne.

QUESTION 6. (1 point) Implémentez une fonction `AfficherParents` qui permet d'afficher les parents d'une personne.

QUESTION 7. (2 points) Implémentez une fonction `AfficherFratrie` qui permet d'afficher la fratrie d'une personne ainsi que sa caractéristique (frère/sœur ou demi-frère/demi-sœur).

QUESTION 8. (2 points) Implémentez une fonction `AfficherFamille` qui, à partir de deux personnes, affiche l'ensemble des enfant du couple et précisnt ceux qui sont issus des deux même parents.