PROJET : Écosystème - Simulation

Exercice 1 (obligatoire) – Tests des fonctions et main

Les fonctions évoquées pendant le TD sont fournies, ainsi qu'un *main* permettant de tester ces fonctions (*main_tests.c*). Un Makefile est également fourni pour créer un exécutable *tests_ecosys*.

1. Il semblerait que les fonctions fournies contiennent des erreurs...
Testez le programme, vérifiez si son comportement est correct et corrigez les erreurs éventuelles en vous appuyant sur les outils de débogage qui vous ont été présentés. Vous pouvez également ajouter des *assert* dans les fonctions pour vérifier que les variables ne sortent pas des valeurs ou intervalles attendus. Une bonne pratique consiste à insérer des *assert* chaque fois qu'un bug est découvert pour éviter qu'il ne se reproduise lors des évolutions ultérieures de votre code.

2. Pour faciliter les manipulations des listes de proies et de prédateurs, écrivez une fonction permettant d'ajouter un animal à la position x, y. Cet animal sera ajouté à la liste chaînée *liste animal*. Plutôt que de renvoyer l'adresse du premier élément de la liste, vous passerez ce pointeur par adresse de façon à pouvoir le modifier directement dans la fonction : l'argument sera donc un pointeur sur liste chaînée, donc un pointeur sur un pointeur sur un Animal…
Vous vérifierez (avec *assert*) que les coordonnées indiquées sont correctes c'est-à-dire positives et inférieures à SIZE X ou SIZE Y qui sont des étiquettes (*#define*) définies par ailleurs.
La fonction aura le prototype suivant : *void ajouter_animal(int x, int y, Animal **liste_animal);*

3. Écrivez une fonction main dans un fichier *main_tests2.c* qui va créer 20 proies et 20 prédateurs à des positions aléatoires, vérifiez leur nombre en faisant appel aux fonctions de comptage vues précédemment et enfin affichez l'état de votre écosystème.
Vous compléterez aussi le Makefile pour créer le programme *tests_ecosys2*.

4. Ecrivez la fonction *liberer_liste_animaux(Animal *liste)* qui permet de libérer la mémoire allouée pour la liste.

5. Complétez la fonction main de *main_tests2.c* pour libérer toute la mémoire allouée.

6. Pour vérifier qu'il n'y a pas de fuite mémoire, il faut utiliser l'utilitaire valgrind. Lancez votre programme avec lui (en tapant valgrind puis votre programme - ex : valgrind ./tests ecosys2).

L'objectif est de n'avoir aucun octet qui soit perdu :
==66002== LEAK SUMMARY:
==66002== definitely lost: 0 bytes in 0 blocks
==66002== indirectly lost: 0 bytes in 0 blocks 7.

Écrivez la fonction permettant d'enlever un élément de la liste chaînée et de libérer la mémoire associée. Comme précédemment, la liste sera passée par adresse. La fonction aura le prototype suivant : *void enlever_animal(Animal **liste, Animal *animal);*
Ajoutez dans votre main la suppression de quelques proies et quelques prédateurs avant la libération de la liste entière, tout en vérifiant que les comptages sont toujours corrects. Vérifiez à nouveau qu'il n'y a pas de fuite mémoire avec valgrind.

Exercice 2 (obligatoire) – Lecture/écriture de fichiers

1. Complétez le main du fichier *main_test2.c* pour écrire l'écosystème créé et le lire dans de nouvelles listes. Vous vérifierez que les écosystèmes lus et écrits sont bien les mêmes. Vous prendrez soin de libérer la mémoire allouée et de vérifier qu'il n'y a pas de fuite mémoire.

Exercice 3 (obligatoire) – Déplacement et reproduction

1. Complétez le main du fichier *main_ecosys.c* pour tester la fonction de déplacement. Vous ne créerez qu'un animal à une position que vous aurez définie et en le déplaçant dans une direction que vous aurez aussi définie. Vérifiez bien la toricité du monde.

2. Complétez le main du fichier *main_ecosys.c* pour tester la fonction de reproduction. Vous mettrez le taux de reproduction a 1 en vérifiant que le nombre d'animaux est bien multiplié par 2 à chaque mise à jour.

Exercice 4 (obligatoire) – Gestion des proies

1. Écrivez une fonction de mise à jour des proies. Cette fonction devra :
— *faire bouger les proies en appelant la fonction bouger animaux;*
— *parcourir la liste de proies :*
— *baisser leur énergie de 1,*
— *supprimer les proies dont l'énergie est inférieur à 0;*
— *faire appel à la fonction de reproduction.*

La fonction aura le prototype suivant : *void rafraichir_proies(Animal **liste_proie, int monde[SIZE_X][SIZE_Y]);*

2. Dans la fonction main du fichier *main_ecosys.c,* vous créerez 20 proies, puis vous écrirez une boucle qui s'arrête lorsqu'il n'y a plus de proies ou qu'un nombre maximal d'itération est atteint (par exemple 200). Dans cette boucle, vous mettrez à jour les proies et

afficherez l'écosystème résultant. Pour que vous ayez le temps de voir l'état de votre écosystème, vous pourrez ajouter des pauses en utilisant la fonction *usleep*.
man 3 usleep pour avoir une description de son fonctionnement et de la façon de l'utiliser.

Exercice 5 (obligatoire) – Gestion des prédateurs

1. Pour gérer les prédateurs, nous aurons besoin d'une fonction qui va vérifier s'il y a une proie sur une case donnée. Ecrivez cette fonction qui aura le prototype suivant :
*Animal *animal_en_XY(Animal *l, int x, int y);*
l est la liste chainée des proies et la valeur renvoyée est un pointeur sur une proie dont les coordonnées sont x et y (la première rencontrée) ou *NULL* sinon.

2. Les prédateurs sont gérés comme les proies. Comme ils utilisent la même structure, les fonctions créer animal, ajouter en tête animal, etc. peuvent être réutilisées telles quelles. Ecrivez la fonction de mise à jour de prédateurs qui respectera le prototype suivant :
*void rafraichir_predateurs(Animal **liste_predateur, Animal **liste_proie);*
Cette fonction est directement inspirée de la fonction de rafraîchissement des proies. Vous pourrez copier-coller celle-ci et faire des modifications. Vous devrez notamment baisser l'énergie d'un prédateur et faire en sorte que, s'il y a une proie située sur la même case qu'un prédateur, elle soit "mangée", permettant ainsi au prédateur de récupérer les points d'énergie de la proie.

3. Modifier votre fonction main (*main_ecosys.c*) pour voir évoluer également les prédateurs. Vous pouvez à présent observer l'évolution de votre petit écosystème et notamment tester différentes valeurs des constantes pour étudier l'impact que cela peut avoir sur votre écosystème.

Exercice 6 (entrainement) – Gestion de l'herbe

Nous pouvons enrichir notre écosystème en ajoutant de l'herbe.Les proies mangent de l'herbe, mais la quantité d'herbe est limitée : si une proie mange l'herbe d'une case, il n'y en a plus et l'herbe met un certain temps à repousser. La quantité d'herbe et le temps de repousse sont modélisés par un tableau à 2 dimensions d'entiers. Si la valeur d'une case est positive ou nulle, il y a de l'herbe, sinon non.

1. Dans le fichier *main_ecosys.c* vous déclarez un tableau statique à 2 dimensions d'entiers de taille *SIZE_X*SIZE_Y* et vous l'initialiserez a 0 dans toutes ses cases.

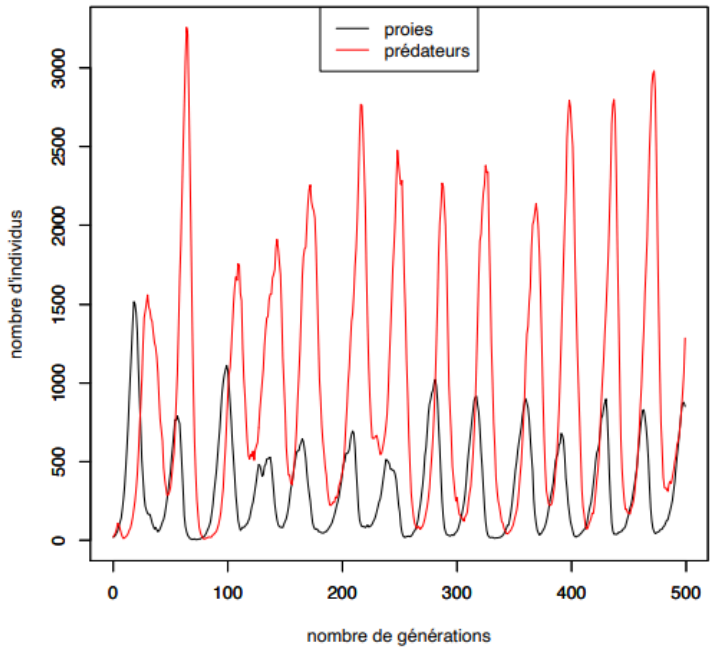
2. A chaque itération de la simulation, la quantité d'herbe de chaque case est incrémentée de 1.
Ecrivez une fonction *void rafraichir_monde(int monde[SIZE_X][SIZE_Y])* qui effectue cette operation.

3. Ajouter la prise en compte de l'herbe dans la fonction de mise a jour des proies. Les proies mangent de l'herbe s'il y en a sur leur case en gagnant autant de points d'énergie qu'il y a d'herbe sur la case. Il faut alors mettre le temps de repousse de l'herbe de la case ou est l'animal à *temps_repousse_herbe* (qui est négatif).

Exercice 7 (approfondissement) – Graphiques de l'évolution des populations

Comme vous l'avez peut-être remarqué, et si vos paramètres le permettent, le nombre d'individus oscille au cours des générations. Nous voulons tracer le graphique du nombre d'individus (proies, et prédateurs si vous les avez simulés) en fonction du nombre d'itérations. Pour cela, il faut écrire dans un fichier à chaque itération une ligne contenant l'indice de l'itération, le nombre de proies, le nombre de prédateurs, séparés par des espaces.

1. Dans le fichier *main_ecosys.c* modifiez votre main pour réaliser cela. Vous pourrez ensuite tracer vos courbes dans un tableur ou avec *gnuplot* en tapant gnuplot puis en tapant
plot "Evol_Pop.txt"˜using 1:2 with lines title "proies"
replot "Evol_Pop.txt"˜using 1:3 with lines title "predateurs"
si vous avez écrit les comptages dans un fichier nommé *Evol_Pop.txt*. Vous devriez obtenir une courbe ressemblant a cela :



2. Faites varier les paramètres du modèle et observez les variations de population.