TP n° 3 : Implantation d'un eco-système (Séance 1)

Le but de cette séance est d'implémenter un ensemble de fonctions sur des listes qui vont permettre de générer, de faire évoluer et d'afficher un éco-système.

Organisation des fichiers :
— Le programme principal est contenu dans le fichier `principal.c`.
— Toutes les fonctions de manipulation et d'affichage des listes chainées sont dans `ecosys.c`. La description de la structure de liste et des constantes utilisées sont dans `ecosys.h`.
— Le fichier `aleatoire.c` contient une fonction qui permet de générer des entiers aléatoires.
Vous ne devez pas changer l'organisation des fichiers.

Notre éco-système est un monde discret (un tore, que nous afficherons comme un rectangle) contenant un certain nombre de cases, identifiées par leurs coordonnées (entières) x et y. Chaque proie (et chaque prédateur) est dans une case donnée et peut se déplacer. A un instant donné, une case peut contenir plusieurs proies et plusieurs prédateurs. La simulation de notre éco-système repose sur plusieurs fonctions que vous allez écrire dans la suite. Les données utilisées pour la simulation sont une liste chaînée contenant les proies et une autre liste chaînée contenant les prédateurs.

```
Nombre de proies (*): 14
Energie torale des proies : 113
Nombre de predateurs (0): 11
Energie torale des predateurs : 109
@=* et 0
+----------+
|        *0|
|0*    *   |
|@*0       |
|    *     |
|0    *0   |
| *      *|
|    0 0   |
|     0   *|
|0    *    |
|   0    * |
+----------+
```

FIGURE 1 – Exemple d'affichage du monde

Les proies et les prédateurs seront représentés par une même structure de données contenant les coordonnées entières x et y, un nombre réel $energie$, un tableau dir de deux entiers dans $\{-1, 0, 1\}$ qui représentera sa direction. Comme nous souhaitons stocker des variables de ce type dans des listes simplement chaînées, la structure contient un pointeur $suiv$ sur cette même structure.

Exercice 1 – Génération de la liste et destruction

Question 1

Ecrire la fonction `PTRANIMAL creationAnimal(int x,int y,float energie)` qui crée un animal et en renvoie l'adresse. Cette fonction renvoie `NULL` en cas de problème. Les deux valeurs de la direction sont initialisées de manière aléatoire.

Question 2

Ecrire la fonction `PTRANIMAL ajouterTete(PTRANIMAL tete, PTRANIMAL ptAnimal)` qui rajoute l'animal pointé par $ptAnimal$ en tete de la liste pointée par $tete$. La fonction retourne l'adresse de la liste chaînée obtenue.

Question 3

Ecrire la fonction `PTRANIMAL initListe (int nbAnimaux)` qui construit une liste chaînée de `nbAnimaux`. Les coordonnées x et y sont tirées alétaoirement entre les valeurs $\{0, \ldots, SIZEX - 1\}$ et $\{0, \ldots, SIZEY - 1\}$. L'énergie est fixée à $INITENERGY$.

Question 4

Ecrire la fonction `int enleverAnimal(PTRANIMAL* ptTete, PTRANIMAL ptAnimal)` qui retire l'animal pointé par $ptAnimal$ dans la liste pointée par $*ptTete$. La fonction retourne 0 si l'animal n'est pas dans la liste (et donc n'a pas pu être retirée), 1 sinon.

Question 5

Ecrire la fonction `PTRANIMAL detruireListe(PTRANIMAL tete)` qui détruit la liste chaînée pointée par $tete$ et retourne $NULL$.

Question 6

Ecrire un programme qui vous permet de tester vos fonctions.

Exercice 2 – Affichage du monde

Le but de cet exercice est d'afficher l'état du monde tel que présenté par la figure 1.

Question 1

Ecrire les fonctions récursives `int compteAnimal(PTRANIMAL tete)` et `float EnergieTotale(PTRANIMAL tete)` qui comptent le nombre d'animalux et l'énergie totale dans la liste pointée par tete

Question 2

Ecrire la fonction `void affiche(PTRANIMAL teteProies, PTRANIMAL tetePredateurs)` qui affiche les proies et les prédateurs tel que présenté dans la Figure 1. Le symbole $*$ signifie qu'il y a au moins une proie dans la case correspondante. Le symbole O correspond à un prédateur. Quand il y a à la fois une proie et un prédateur dans une case, on utlise le symbole @.

Pour cette fonction, on pourra utiliser une variable locale qui est une matrice de $SIZEX \times SIZEZ$ caractères à remplir en fonction des deux listes et à afficher.

Question 3

Ecrire un programme qui vous permet de tester vos fonctions.

Exercice 3 – Déplacements et reproduction

Les mouvements animaux (proies ou prédateurs) seront gérés de la façon suivante : chaque animal dispose d'une direction indiquée dans le champs dir, qui est un tableau de deux entiers. Il indique la direction suivie sous la forme de deux entiers valant -1, 0 ou 1. Ces valeurs indiquent de combien les coordonnées de l'animal doivent être décalées. Considérant que la première case de la prairie est en haut à gauche, une direction de $(1, -1)$ correspond, par exemple, à un mouvement vers la case en haut et à droite, une direction de $(0, 0)$ correspond à un animal immobile.

De plus, le monde sera supposé torique, c'est- à-dire que si l'animal essaie d'aller en haut alors qu'il est sur la première ligne, il se retrouvera automatiquement sur la dernière ligne. De même si un animal essaie d'aller à droite alors qu'il est sur la dernière colonne de droite, il réapparait sur la même ligne et sur la colonne la plus à gauche.

Question 1

Ecrivez une fonction `PTRANIMAL deplacement (PTRANIMAL tete, float coutEnergie)` permettant de faire bouger les animaux contenus dans la liste chaînée passée en argument $tete$. De plus, chaque animal consomme une énergie $coutEnergie$ à chaque tour et tous les animaux dont lénergie est strictement négative sont éliminés de la liste (et de la mémoire).

Question 2

Ecrivez une fonction `PTRANIMAL reproduction(PTRANIMAL tete, float probaReproduction)` qui permet de gérer la reproduction des animaux. Vous parcourez la liste passée en paramètre et, pour un animal $cour$ vous ajouterez un nouvel animal à la même position que $cour$ avec une probabilité $probaReproduction$. L'ajout se fera en tête. Un animal qui vient de naître ne sera pas considéré par votre boucle et ne se reproduira donc pas.

Question 3

Testez vos fonctions en ne créant qu'un animal à une position que vous aurez définie et en le déplaçant dans une direction tirée aléatoirement. Vérifiez bien la toricité du monde. Testez aussi votre fonction de reproduction. Pour ralentir le programme entre deux mouvements, vous pourrez effectuer l'appel `usleep(1000000)`.