Ana Carolina da Silva Carvalho de Sousa
Carlos Adir Ely Murussi Leite

1 Question 1

1.1

La fonction LireEnvironnement lit un fichier au format .dat, l'un des trois fichiers disponibles (monde1.dat, monde2.dat ou monde3.dat) et lit les paramètres suivants:

(a) largeur et à la hauteur réelles du monde

(b) position (abscisse et ordonnée) de la fourmilière;

(c) Les valeurs possibles pour le monde qui peuvent être OBSTACLE, VIDE ou bien une valeur sont comprises entre VIDE et NOURRITURE MAX.

Ainsi, les valeurs de ces variables ont été stockées dans une variable "t_monde" et la fonction renvoie cette variable.

1.2

La fonction principale a été lancée en lisant un fichier monde.dat et en le stockant dans une variable t_monde appelée "World". Avec cela, vous pouvez utiliser la fonction InitAffichage pour vérifier si l'exécution du programme est possible en considérant la hauteur et la largeur du monde de lecture.

Il faut utiliser les fonctions "Mise À Jour Environnement" et "Affichage" pour rafraîchissement de l'écran.

1.3

Pour que les tests soient effectués, il était également nécessaire d'initialiser les fourmis. Pour cela, il est nécessaire de créer une structure t_fourmi contenant les caractéristiques d'une fourmi.

Le type t_fourmi a été défini par un enregistrement avec trois caractéristiques:

(a) Position X dans le Environnement

(b) Position Y dans le Environnement

(c) Mode: trouvé de la nourriture(0) ou recherche(1) de nourriture

(d) Dir: Direction la fourmi se promène dans l'environnement. Entre -4 et 4.

Algorithime de la structure t_fourmi:

```
Enregistrement
    Entier X;    // Position de la fourmi dans la abscisse;
    Entier Y;    // Position de la fourmi dans la ordonnee;
    Entier Mode; // Mode: trouvé de la nourriture(0) ou recherche(1) de nourriture
    Entier Dir;   // Direction que la fourmi marche dans l' enviromment. Entre -4 et 4.
Fin Enregistrement
```

1.4

La fonction DeplaceFourmi a pour but, lors de la réception d'une fourmi au format "t_fourmi" et d'un environnement au format "t_monde", de déplacer celle-ci de manière aléatoire dans cet environnement. Pour cela, une convention de pilotage a été choisie. Cette convention consistait à définir 8 directions, qui seraient tirées au hasard par la fonction de Nalea, et à partir de là la fourmi change de trajectoire.

La convention de changement de direction a été donnée par:

5	6	7
4		0
3	2	1

L'algorithme de cette fonction est donné par:

```
Variables
    Entier rot, dx, dy;
Debut
    Faire
        rot <- nalea (8);
        choix selon:
            rot = 0:
                dx <- 1
            rot = 1:
                dx <- 1
                dy <- -1
            rot = 2:
                dy <- -1
            rot = 3:
                dx <- -1
                dy <- -1
            rot = 4:
                dx <- -1
            rot = 5:
                dx <- -1
                dy <- 1
            rot = 6:
                dy <- 1
            rot = 7:
                dx <- 1
                dy <- 1
        fom choix selon

    //Cette partie a été ajoutée plus tard, uniquement pour résoudre le problème de marcher sous les

    Tant que non PositionPossible(f.x + dx, f.y + dy, monde)
    f.x <- f.x + dx
    f.y <- f.y + dy
    renvoyer f
```

1.5

L'initialisation des fourmis a été traitée en 1.2

1.6

L'initialisation des fourmis a été traitée en 1.2

1.7

Premièrement, seule une structure de déplacement de la position des fourmis a été utilisée au hasard. Avec cela, les fourmis ont commencé à marcher sur les obstacles et à sortir de l'environnement. Pour résoudre ce problème, nous avons utilisé la fonction intégrée à la question 2: PositionPossible.

2 Question 2

2.1

La fonction PositionPossible obtient une certaine position, donnée par les coordonnées X et Y, un environnement au format t_monde et renvoie si cette position peut être occupée par une fourmi (true si possible et false si ce n'est pas possible). C'est-à-dire cette position est possible si elle est dans la carte et si elle ne représente pas un obstacle.

```
Fonction PositionPossible
fonction resultat <- PositionPossible(x, y, monde)
parameters
```

```
    Entier x:
    Entier y:
    t_monde monde:
resultats
    boolean resultat
Algorithme
    Debut
        resultat <- faux
        si (x < monde.L et x >= 0 et y >= 0 et y < monde. H)
    si ( monde.mat[x][y] est different de -1)
    resultat <- true;
fin si
    fin si
    renvoyer resultat
    Fin
```

2.2

La fonction donnée est completée et s'appelle DeplaceFourmi_Better dans le code.

2.3

Le premier changement observé a été que les fourmis ont cessé de quitter l'environnement et ont cessé de marcher sous les obstacles, c'est-à-dire qu'elles ont commencé à marcher uniquement dans les positions possibles de la carte.

Le deuxième changement observé est qu'avec le vecteur de pondération, le poids le plus élevé étant attribué à la direction 0, c'est-à-dire qu'on continue dans la même direction, on peut voir que les fourmis tendent à effectuer une trajectoire plus droite qu'auparavant.

```
Enregistrement
    Entier X;    // Position de la fourmi dans la abscisse;
    Entier Y;    // Position de la fourmi dans la ordonnee;
    Entier Mode; // Mode: trouvé de la nourriture(0) ou recherche(1) de nourriture
    Entier Dir;   // Direction que la fourmi marche dans l' enviromment. Entre -4 et 4.
Fin Enregistrement
```

2.4

3 Question 3

3.1

Fonctions terminées dans le fichier fourmi.cpp.

3.2

La fonction améliorée DeplaceFourmi (appelée DeplaceFourmi_Better) se trouve dans le fichier fourmi.cpp.

3.3

La performance du programme a été mesurée par une variable (nombre_it) trouvée dans main.cpp. Cette variable a été initialisée à 0 et augmente la valeur d'une unité à chaque mise à jour du programme. La variable est imprimée à l'écran lorsque la norriture de l'environnement est terminée.

Pour le programme Question 3, c'est-à-dire sans implémentation de phéromones, le nombre d'itérations imprimées était de 4263.

4 Question 4

4.1

Afin de pouvoir représenter le niveau de phéromone en chaque point de l'environnement, une variable t_matrice a été initialisée dans la fonction principale. Ce tableau a été initialisé avec 0 dans chaque élément.

La fonction d'évaporation des phéromones a également été assemblée et, à chaque mise à jour du programme, évapore une unité de tous les éléments de la matrice des phéromones.

4.2

Ensuite, pour chaque mise à jour du programme (effectuée dans la fonction principale), pour toutes les fourmis retournant en mode fourmilière, les éléments de ce tableau de phéromones sont modifiés en fonction de la position occupée par chaque fourmi retournante. À chaque mise à jour, il était testé si le niveau maximum de phéromone n'était pas dépassé (100).

4.3

Premièrement, nous avons trouvé la direction autour de la fourmi qui contenait la plus grande quantité de ferormonium via la fonction get_dir_phero. Cette fonction reçoit la matrice de phéromones et une fourmi au format t_fourmi, juste pour obtenir leur position, et renvoie une valeur de 0 à 7, indiquant la position contenant le plus grand volume de phéromone.

Avec la direction découverte, nous avons redéfini le vecteur de pondération selon la formule tauxPheromone * (abs (directionOpposée)). Autrement dit, dans la direction de la phéromone supérieure, le poids est nul (car on ne veut pas suivre une fourmi qui retourne à la fourmilière), et un poids de 1, 2, 3, 4 $\times$ valeur de phéromone pour les autres directions.

4.4

Pour le programme Question 4, c'est-à-dire avec implémentation de phéromones, on peut constater que le nombre d'itérations nécessaires pour achever la norriture diminue considérablement.