Projet de programmation en C
Invasion de pucerons !

Marine Djaffardjy (marine.djaffardjy@universite-paris-saclay.fr)

Contexte et règles de simulation

Cet été, vous avez décidé de planter des tomates dans votre potager. Vos tomates sont très épanouies, mais des intrus très intéressés par vos superbes tomates font irruption : des pucerons. Ils dévorent toutes vos tomates ! Pour protéger vos plantes, vous trouvez une solution et introduisez un prédateur des pucerons : les coccinelles.

Votre objectif va être de simuler l'évolution de la population de pucerons et de coccinelles dans le potager.

Votre potager sera représenté par une matrice sur laquelle vos insectes évolueront, de taille 30×30.

Lors de l'initialisation de cette matrice, chaque case contient des tomates mûres. Vous aurez également 20 pucerons et, si vous avez des coccinelles, 10 coccinelles (sinon zéro) répartis au hasard sur votre grille.

Au départ, chaque case ne contiendra donc qu'une seule entité. Ensuite lors de la simulation, il pourra y avoir un insecte de chaque type au début d'un tour, mais il n'y en aura plus qu'un à la fin du tour.

Projet à réaliser en binôme

Pucerons

Déplacement Les pucerons ont pour objectif de manger le plus de tomates possibles. Ils se déplaceront donc dans une direction donnée jusqu'à ce qu'ils ne trouvent plus de tomates, auquel cas ils se dirigeront dans une case attenante au hasard contenant une tomate. Si ils n'en trouvent pas, ils se déplaceront sur une case au hasard.

Reproduction Si les pucerons mangent pendant 5 tours consécutifs, ils peuvent se reproduire.

Durée de vie Les pucerons survivent, sauf s'ils sont mangés, 10 tours.

Affichage Pour l'affichage en console, le symbole d'un puceron sur une case va dépendre de sa direction :
- vers la gauche $<$
- vers la droite $>$
- vers le haut $\wedge$
- vers le bas $\vee$
- en diagonale NO ou SE $\backslash$
- en diagonale NE ou SO $/$

Coccinelles

Les coccinelles ont pour objectif de manger les pucerons. Elles peuvent détecter des pucerons dans un rayon de trois cases autour d'elles. Elles se déplaceront, si elles détectent un puceron, dans sa direction. Si elles ne trouvent pas de puceron pendant quelques tours, elles changeront de direction au hasard. Autrement, elles iront tout droit.

Reproduction Si les coccinelles mangent 3 pucerons, elles peuvent se reproduire.

Durée de vie Les coccinelles survivent, quoi qu'il arrive, 20 tours.

Affichage Pour l'affichage en console, le symbole d'une coccinelle sur une case va dépendre de sa direction :
• vers la gauche $<$
• vers la droite $>$
• vers le haut $\wedge$
• vers le bas $\vee$
• en diagonale NO ou SE $\backslash$
• en diagonale NE ou SO $/$

Le potager

Votre potager est une grille de taille 30*30. Chacune de ces cases contient un plant de tomate pouvant faire pousser au maximum une tomate. Lorsqu'un puceron se trouve sur une case contenant une tomate, il la mange, mais quand une tomate disparaît, elle repousse ! Il faut 20 tours à votre tomate pour repousser entièrement.

Affichage Pour l'affichage en console, vous devrez afficher plusieurs stades de maturité de votre tomate :

- **.** La tomate a été mangée, il n'y a pas de tomate (tour 1 à 4 après la visite du puceron)
- **o** La tomate repousse et est encore verte (tours 5 à 19 après la visite du puceron)
- **O** La tomate a fini de pousser et est mûre (et juteuse !) (à partir du tour 20 après la visite du puceron)

Pour l'affichage en fenêtre, utlisez un gradient de rouge pour figurer les tomates.

Implémentation

Vous avez deux niveaux de simulation :

- **Niveau 1** : Simulation de l'invasion de pucerons dans le potager. (seulement les pucerons)
- **Niveau 2** : Introduction des coccinelles. (pucerons + coccinelles)

Déroulement d'un tour de simulation *Pour le niveau 1, ignorer les actions et interaction avec les coccinelles.*

Vous commencerez par mettre à jour l'état des tomates.

Lors d'un tour de simulation, c'est d'abord l'ensemble des pucerons qui devront être déplacés. Viendra ensuite le tour des coccinelles. Ensuite, si deux insectes (puceron et coccinelles) sont sur la même case, la coccinelle mange le puceron. Si le puceron a survécu au tour, il doit :

- Se nourrir : si il se trouve sur une case contenant des tomates, même non mûres
- Se reproduire : si il a mangé pendant 5 tours, le puceron se reproduit et un deuxième puceron apparaît sur une case attenante au hasard.
- Mourir : si le puceron a atteint l'âge de 10 tours, il disparaît.
- S'orienter : (si le puceron est toujours vivant) ne peut pas continuer dans sa direction initiale (si il rencontre un obstacle ou n'a plus de nourriture dans sa direction initiale), il doit se réorienter. Il doit alors choisir une direction au hasard dans une des cases attenantes, en priorité celles contenant de la nourriture.

Vient le tour des actions des coccinelles :

- Se reproduire : si la coccinelle a mangé trois pucerons, elle se reproduit et une nouvelle coccinelle apparaît dans une case attenante au hasard (vide).
- Mourir : Si la coccinelle a 20 tours d'âge, elle disparaît.
- S'orienter : Si la coccinelle a trouvé et mangé un puceron (ou perdu la trace d'un puceron), elle doit se réorienter. Elle se dirige vers une case attenante au hasard selon

Aide à la modélisation

Étape 1

Vous aurez besoin de 3 structures: Coordonnées, Coccinelle et Puceron (ou 2 si vous n'avez pas de coccinelles).

Étape 2

Afin de pouvoir mettre à jour l'ensemble des pucerons avant l'ensemble des coccinelles, créez deux tableaux : un stockant les pucerons, un pour les coccinelles. À chaque tour, vous parcourrez une seule fois chacun de ces tableaux, et non 3 fois l'écosystème entier.

Si un insecte meurt, il devra être retiré du tableau. Lorsqu'un insecte naît, il devra être ajouté au tableau. Soyez astucieux: lors de la suppression d'éléments dans le tableau, ne translatez pas l'ensemble des éléments suivants pour combler le vide : comblez-le en allant chercher le dernier.

Modularité Veuillez à découper judicieusement votre projet.
Vous devriez avoir des fichiers séparés pour la gestion:
- de l'initialisation
- de la simulation
- des cases
- de l'affichage en console
- de l'affichage en fenêtre graphique si vous allez jusque là

Bonus

- si vous "supprimez" les bords, au sens que le bord gauche est lié au bord droit, et celui supérieur à l'inférieur. Exemple dans ce cas de figure: un insecte situé contre le bord gauche qui se dirigerait sur sa gauche, rentrerait sur le bord droit de la carte.
- si vous réalisez une interface graphique. Pour cela, utilisez la librairie SDL, et les images que je vous mets à disposition pour symboliser les entités. Laissez à l'utilisateur le choix d'afficher la simulation en console ou en mode graphique.

1 Questions

Lorsque vous ferez tourner vos simulations, vous pourrez faire certaines observations sur vos tailles de populations :

- **Niveau 1** : à partir de combien de tours les pucerons ont mangé 75% des tomates du potager (plus que 15% de tomates mûres) ? Qu'observez vous sur la population de pucerons au bout de 10,50,100 tours ?
- **Niveau 2** : à partir de combien de tours les coccinelles ont mangé tous les pucerons ?
- **Bonus** : Quels paramètres faudrait il changer pour que la population de pucerons ne disparaisse pas ? Donnez des valeurs pour ces paramètres. Qu'observez vous sur les tailles des populations de pucerons et de coccinelles ?

Évaluation

Vous m'enverrez par mail une archive contenant votre projet (les fichiers .c, .h et le makefile permettant de compiler) ainsi qu'un rapport pour répondre aux questions et expliciter jusqu'où vous avez traité le sujet, les potentiels choix réalisés, les difficultés rencontrées...
Ce projet est un travail en binôme. La date limite du rendu du projet est fixée au samedi 06 janvier 2023 23h59.
Votre projet doit pouvoir être compilé et exécuté sous Unix.

Pour ceux qui connaissent ou souhaitent se former aux gestionnaires de version permettant un travail collaboratif, vous pouvez utiliser GitHub ou GitLab et m'adresser pour rendu le lien vers celui-ci (et non l'archive).

En plus du fonctionnement du programme, votre code sera évalué selon les critères suivants:

- Le code doit être lisible (indentation, espacement, longueur des lignes).
- Le nom des variables, fonctions et structures doit être pertinent et respecter les règles de formatage.
- Utilisation pertinente de constantes.
- Chaque fonction doit se charger d'une seule tâche bien définie et spécifiée sans ambiguïté dans le commentaire qui accompagne la fonction.
- Les passages importants du code doivent être commentés de sorte à ce qu'on puisse comprendre vos raisonnements.
- Le découpage modulaire doit être pertinent : il regroupe les fonctions par thématique.
- Votre Makefile devra invoquer l'option -Wall pour le compilateur gcc. Tous les avertissements seront pris en compte dans l'évaluation.