Logique de developpement du test 5 s'appliquant sur une grille avec le déplacement d'un personnage

Victor VITCHEFF

April 6, 2022

Contents

Introduction

Dans ce fichier, je vous présente la logique de développement pour le test 5. Ce test représente le jeu d'un personnage qui se trouve sur la case d'une grille à l'instant t=0. Il doit ensuite atteindre une case noté 'E'. A chaque pas de temps, il peut avancer d'une case sur une des cases voisines mais du feu se propage sur la grille. Le feu se propage à chaque pas de temps sur toutes les cases voisines peu importe la valeur de la case situé à côté de lui. Le personnage ne peut pas avancer s'il est à coté d'un arbre représenté par le caractère '♯'. Pour cette logique de développement, je vais d'abord vous présenter l'analyse du besoin, puis je vous présenterai l'architecture de ma solution et enfin je vous présenterai mon plan de validation de la solution.

I Analyse du besoin

L'objectif de ce projet est de rendre une solution au problème proposé. On doit trouver un chemin pour que le personnage arrive en vie. Le programme retourne un exit_code(0) si c'est un succès et un exit_code(1) si ce n'est pas un succès. C'est donc un problème de décidabilité. S'il y a succès, on écrit aussi le chemin. Je souhaite répondre à ce problème pour ensuite être séléctionné comme stagiaire au sein de votre entreprise.

La durée de mon projet a été très mal géré. J'ai tout de même essayé de faire le plus rapidement possible. Ayant attrapé la grippe durant le mois de mars, je n'ai pas pu être efficace. J'ai aussi eu beaucoup de projets à rendre ce qui m'a rendu la tâche difficile à réaliser. Tout de fois, je vous ai notifié ma fin de projet à chaque fois.

Je n'ai pas réalisé pliusieurs livrables car le projet fut court. Le livrable que j'ai envoyé est celui qui contient le rapport, la logique de développement ci-joint ainsi que les codes. J'ai décidé de n'en envoyer qu'un par soucis de simplicité.

Aucune interface technique ne m'a été imposé, seuleument un choix entre différents langages. J'ai donc choisi le langage C car c'est celui pour lequel je me sentais le plus à l'aise au moment de commencer le test.

II Architecture

II.1 Stratégie de développement

Je vais vous expliquer comment j'ai procédé pour arriver au résultat final de ce test. Le besoin s'articule en 3 phases. La première phase est de récupérer la grille.txt. La deuxième fut de récupérer les informations importantes de la grille (comme la position du personnage à l'instant t=0) et la dernière est de renvoyer l'exit_code correspondant et calculer le chemin s'il le faut. C'est donc ces 3 phases que j'ai décidé de réaliser pour trouver une solution au besoin.

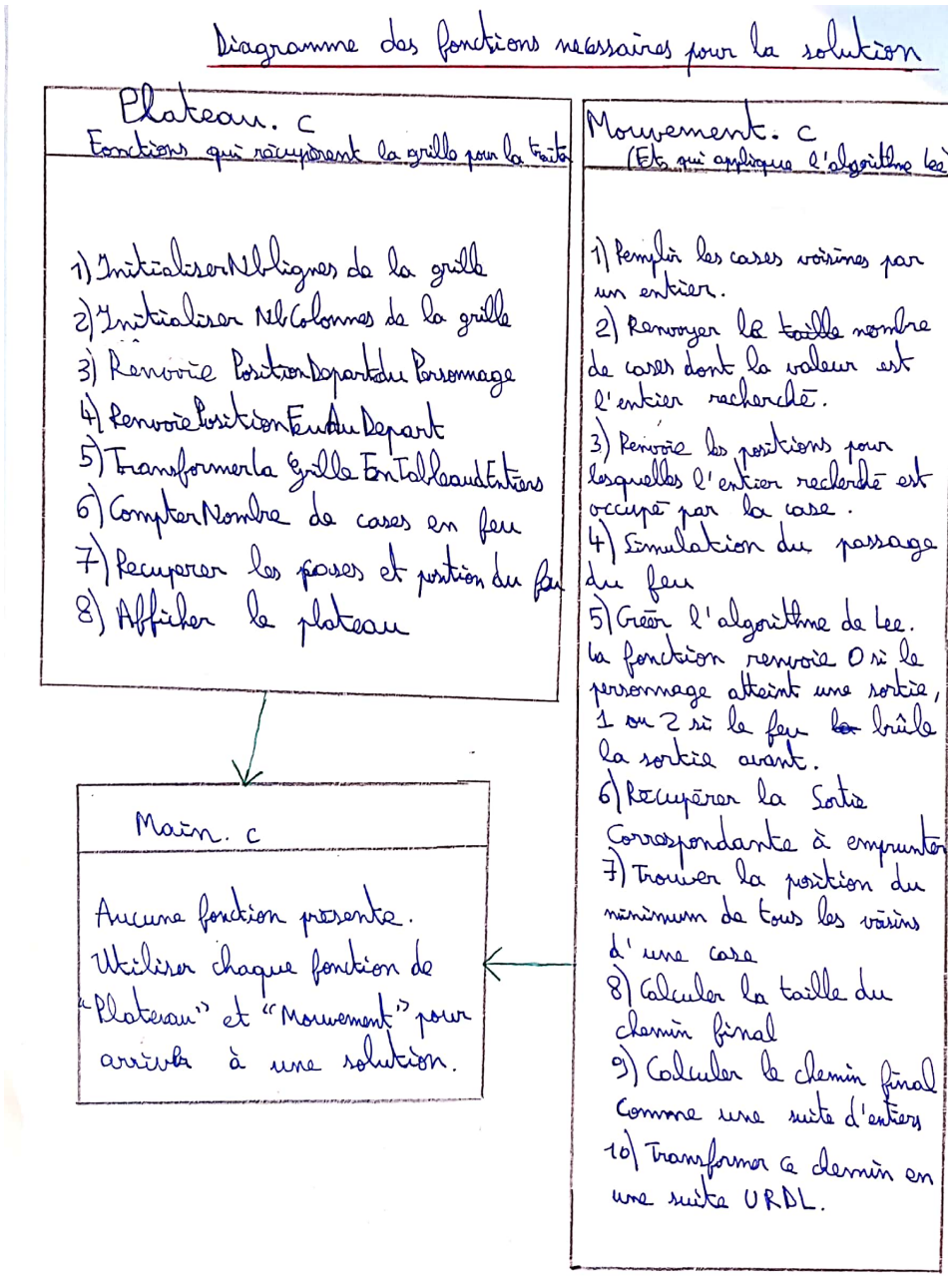
J'ai ensuite procéder au choix du langage. J'avais fini le test 2 avec le langage C. Donc je me sentais à l'aise avec le langage. La deuxième raison qui m'a poussé à choisir le langage C est que la plupart des jeux que j'ai programmés lors de mes deux premières années de prépa intégrée étaient en C. J'ai par exemple programmé un jeu de la vie ainsi que le jeu d'isola en C. Le jeu que vous m'avez proposé pour ce test se rapproche beaucoup du jeu de la vie. C'est pour cette raison que j'ai eu envie de programmer dans ce langage.

Je voulais attirer un point sur le fait que dans mon programme j'utilise beaucoup de tableaux dynamiques. En effet, j'ai représenté le chemin par un tableau, ce chemin change en fonction de chaque grille par exemple. Et je suis conscient qu'il est plus diffcile de créer un tableau dynamique en C. En revanche, comme je suis plus à l'aise dans ce langage, je ne trouve pas que c'est une énorme difficulté de créer un tableau dynamique en C. Il suffit de rajouter une boucle à chaque fois pour calculer la taille du tableau puis ensuite le rendre dynamique grâce à la fonction malloc. J'ai donc décider d'utiliser ce langage, je juge que c'est le langage où je suis le plus efficace pour programmer ce jeu.

II.2 Architecture fonctionnelle

Vue logique

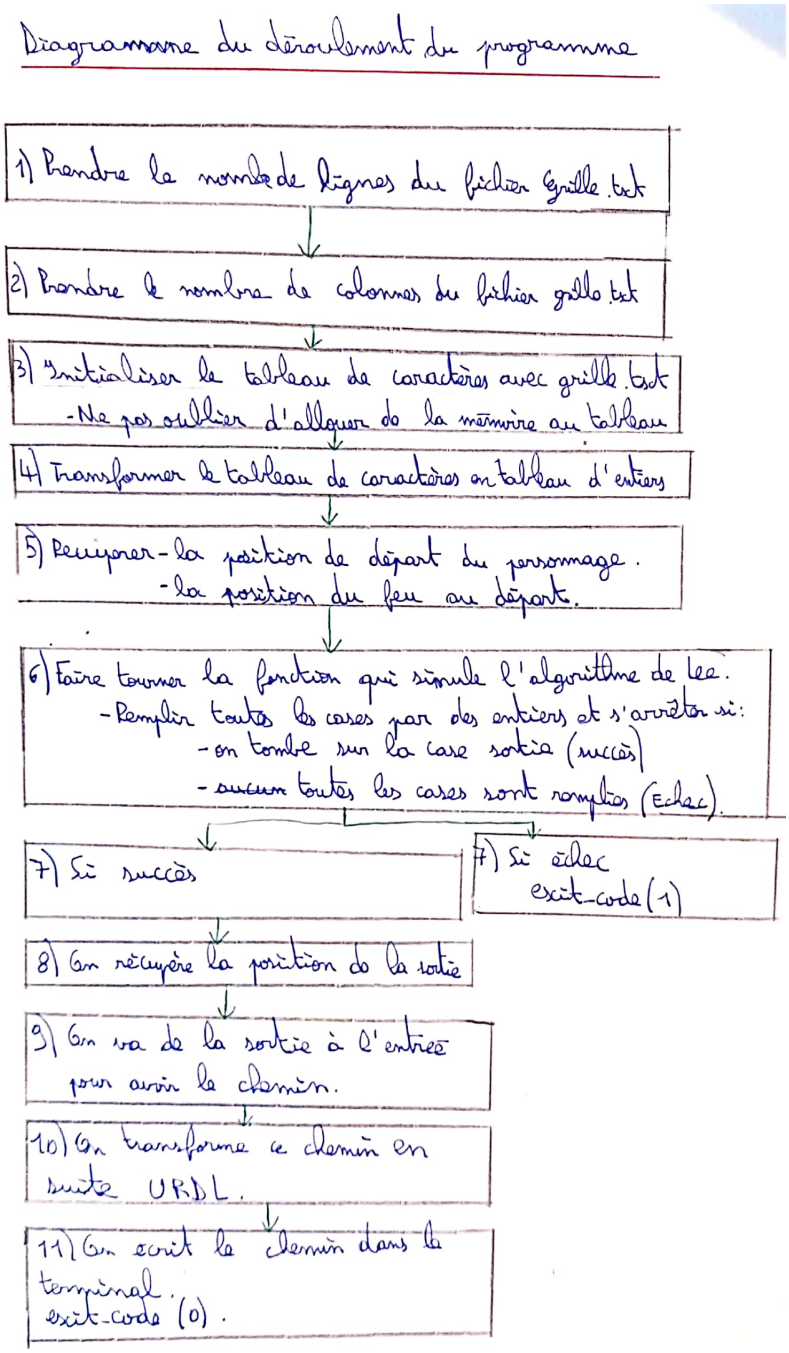
Voici le diagramme présentant l'organisation fonctionnelle du système.



Chaque numéro représente une fonction présente dans le fichier (soit 'plateau.c' soit 'mouvement.c').

Vue de processus

Voici le diagramme présentant le séquencement des opérations du système.



Ce diagramme représente le déroulement de ce qu'il se passe dans le fichier "main.c" à chaque fois qu'on le lance.

II.3 Architecture technique

Vue de réalisation

On cherche à viser toutes les machines. En effet, toute machine dotée du langage de programmation C peut compiler et exécuter le programme. Aucun logiciel n'a été modifié, j'ai seulement utilisé le langage C. Les sources de données utilisés sont des fichiers de nature ".txt" que l'on extrait facilement avec une commande du langage C. Aucun fichier de configuration n'est nécessaire pour faire tourner le programme. Il suffit simplement de posséder le langage C.

Contraintes de réalisation

Beaucoup de fonctions de mon code dépendent d'autres fonctions. Donc en modifiant certains éléments, cela peut influer sur une une grande partie du code. Je n'ai senti aucune difficulté à ne pas respecter les règles de codage. Et elles ont été respecté dans le code que je vous propose. Le respect du codage est tout à fait possible. De plus, comment le langage C est modulable, on peut facilement créer beaucoup de fichiers si l'un des fichiers est trop long.

III Plan de validation

On attend que le programme nous dise si le personnage peut arriver à la position de sortie et si oui par quel chemin. On attend ce résultat pour n'importe quelle grille rentré en paramètre qui respecte les conditions de départ (comme par exemple case terrain libre ='.'). Les composants à valider sont celles de savoir si le code ne renvoie pas toujours un mauvais résultat. Pour trouver le résultat, on peut tester avec les jeu de données en entrée. Mais pour vérifier que c'est la bonne solution, il faut aussi tester à la main voir s'il existe un chemin mais aussi voir si le chemin donné est cohérent. En effet, je n'ai pas de protocole de validation de la grille. Je ne sais pas si la grille est un succès pour le personnage ou un échec. Il convient de noter que le critère de succès est le personnage arrive sur une sortie sans brûler et l'echec le personnage n'arrive pas sur une sortie et brûle.

Je pense que la meilleure stratégie pour valider la solution du problème est dans cette situation, la matrice de confusion. Mais pour cela, il nous faut un ensemble de grille avec leur solution :

- Si la solution est un succès, alors la grille sera muni de tous les chemins allant de l'entrée à la sortie et le booléen associé à la grille sera 1.
- Si la solution est un échec, alors le booléen associé sera 0.

En fonction des données de la grille, on lance le programme C comme décrit dans le rapport et on compare le résultat prédit par le programme et le booléen associé à la grille. On regarde aussi si le chemin renvoyé par le programme est présent parmis les chemins de la grille si la grille est un succès. Le tableau aura cette forme :

	$Prédiction = 0$	$Prédiction = 1$
$Réalité = 0$	TN	FP
$Réalité = 1$	FN	TP

Chaque case correspond à un résultat. Je décris cela ici :

- TN= True Negative, c'est le nombre de réalisations qui ont été prédites échec et qui en réalité sont un échec.
- TP = True Positive, c'est le nombre de réalisations qui ont été prédites succès et qui sont véritablement un succès.
- FN = False Negative, c'est le nombre de grilles que mon programme a prédits fausse alors qu'elles sont vraies.
- FP = False Positive, c'est le nombre de grilles prédites vraies alors que celle ci sont fausses.

A partir de cette matrice de confusion, on peut calculer une sorte d'indice de confiance que l'on note I. Il convient aussi de noter N qui est le nombre total de grilles que l'on a testé. Celui correspond à :

$$I = \frac{TN+TP}{N}$$

Si cette indice de confiance est supérieure à 95%, alors on considère que l'on peut garder le programme car celui est assez précis. Pour les 4 grilles que j'ai obtenu, voici la matrice de confusion que l'on obtient ainsi que l'indice de confiance.

	$Prédiction = 0$	$Prédiction = 1$
$Réalité = 0$	0	0
$Réalité = 1$	0	4

$$\Rightarrow I = \frac{0+4}{4} = 1 = 100\%$$

On remarque que l'indice de confiance est au plus haut possible. Cela signifie que notre solution est validée. Il convient aussi pour valider de manière plus précise la solution en langage C d'augmenter le nombre de grille à tester par le programme. En effet, l'indice de confiance sera de cette manière beaucoup plus précis.

Conclusion

Pour conclure, ma stratégie de developpement a été une réussite puisque l'on observe sur le plan de validation que la solution que j'ai proposé en langage C permet de résoudre le problème proposé. J'ai eu des difficultés à analyser le besoin réel car je ne sais pas si étudier la trajectoire de ce personnage peut vraiment avoir un impact ensuite par la suite sur certains logiciels ou certains problèmes. J'espère que cette stratégie de développement me sera utile par la suite pour pouvoir travailler dans votre entreprise.