Rapport du projet 19°

Quel taquin !

BOUGHATANE Amina
NGOIE-MBAYI Jonathan

Avril 2019

TABLE DES MATIERES

Introduction

- Présentation générale :

Dans le cadre de notre projet, nous avons choisis de réaliser le projet numéro 19, Il consiste à programmer en C le jeu de Quel taquin ! Jeu sur un damier de 4 x 4 cases comportant 15 pions numérotés de 1 à 15. Ces pions ne peuvent se déplacer sur le plateau que par glissement dans la seule case vide à un moment donné. Le jeu consiste à remettre les pions dans l'ordre numérique, Une adaptation de ce jeu consiste à disposer des pièces de taille variée (par exemple des pièces de taille 1 x 1 et des pièces de taille 1 x 2) et de les amener à leur position.

Cahier des charges

Notre première démarche a été de se représenter les différentes étapes nécessaires à la bonne exécution du programme :

- Définir nos deux tableaux (initial et final).
- Fonction pour afficher notre plateau de 16 cases.
- Fonction pour enregistrer notre case vide.
- Proposer des mouvements en déplaçant la case vide.
- Faire des limites pour ne pas sortir de la grille.
- Comparer les deux tableaux pour savoir si le joueur à gagner.
- Définir un tableau (non visible par le joueur) pour distinguer les cases paires (1*1) et impaires (1*2) et ainsi empêcher le joueur à faire des mouvements impossibles.
- Pouvoir quitter le jeu à n'importe quel moment.
- Créer une sorte de menu d'accueil qui explique les règles du jeu pour éviter les répétions et les printf dans la boucle.
- Proposer des mouvements pour gagner à chaque nouvelle situation (Pas réussi).
- Facultatif : Compter le nombre de mouvement (un score).
- Et ainsi de suite…

Cela nous a amené à définir des types spéciaux :

- gridSize 4 : La taille des tableaux à deux dimensions
- int initialGrid : Le tableau initial
- int finalGrid : Le tableau final.
- int backGrid : Le tableau des cases spéciales.
- struct Compartement : Une structure pour enregistrer les coordonnées d'une case.
- int keepPlaying : Variable pour arrêter ou continuer le jeu.

Et à créer des fonctions, dont voici les principales :

- void printGrid() ;
- void initBack() ;
- void saveEmptyCompartment() ;
- int compareGrid() ;
- void movement() ;

Le guide du programmeur

- void printGrid(int grid[gridSize][gridSize]);

La première fonction est printGrid qui nous a permis d'afficher notre plateau de jeu elle prend en paramètres un tableau à deux dimensions de taille 4 et nous affiche tous les éléments de notre tableau case par case en parcourant les lignes et les colonnes avec deux boucles for, et en arrivant à la case contenant le chiffre 0 nous renvoie un « espace vide » qui deviens notre case vide.

- void saveEmptyCompartment(int grid[gridSize][gridSize]);

saveEmptyCompartment à pour paramètres un tableau à deux dimensions de taille 4 elle sert à trouver la position de notre case vide dans le tableau et l'enregistrer grâce à notre structure Compartment et l'utiliser dans la fonction movement et pouvoir modifier ses coordonnées à chaque tour de jeu.

- int compareGrid(int grid1[gridSize][gridSize], int grid2[gridSize][gridSize]);

La fonction compareGrid est un test, elle prend en paramètre deux tableaux de deux dimensions taille 4 et les compare en parcourant les case à l'aide d'une boucle for, renvoie 1 si les deux tableaux sont identiques sinon elle renvoie 0. Cette fonction est importante pour savoir si le joueur à gagner, si c'est le cas compareGrid renvoie 1 et notre variable keepPlaying s'initialise a 0 donc on sort de la boucle while de notre fonction movement ce qui arrête le jeu. Sinon le joueur peut continuer à jouer.

- void initBack(int grid[gridSize][gridSize]);

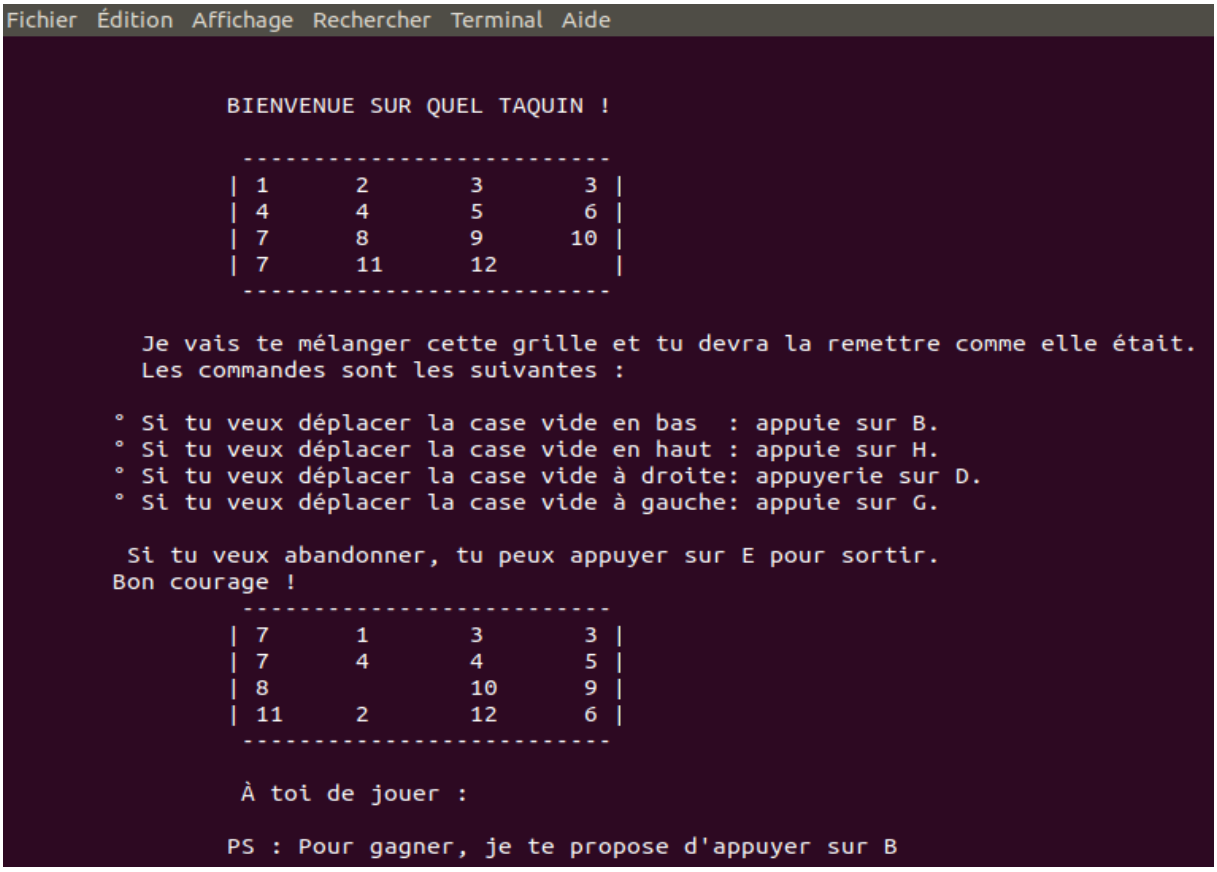
Après de longues recherches et de réflexions on a pu trouver une fonction qui nous a permis de traiter le cas de pièces de taille 1*2, initBack prend en paramètre un tableau de deux dimensions et remplie ce tableau avec quatre types d'élément (0 / 1 / 2 / 3), pour distinguer les cases paires verticalement/horizontalement des cases impaires.

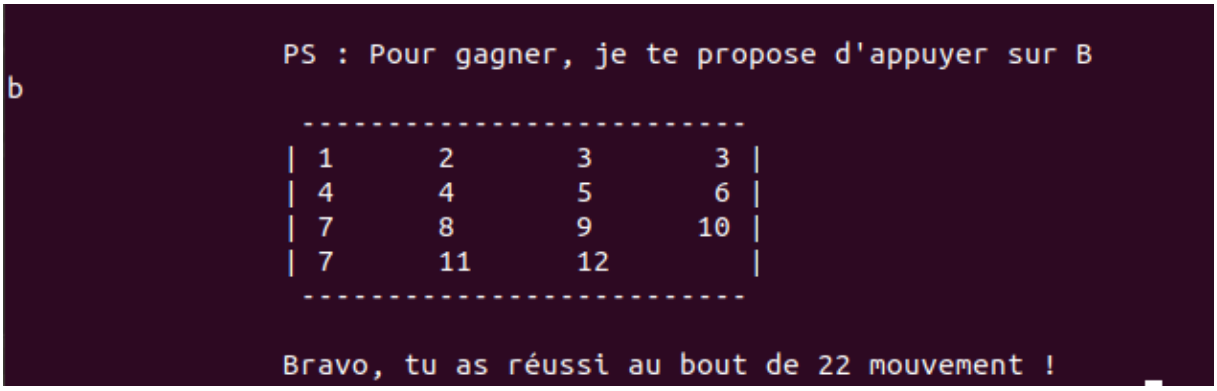
Exemple :

initialGrid				⇔	initBack			
7	1	3	3		3	0	1	1
7	4	4	5		3	2	2	0
8		10	9		0		0	0
11	2	12	6		0	0	0	0

- void movement() ;

movement est la fonction principale, on commence par déclarer deux variables, mov et saveEmptyCompartment qui contient la case vide du début.
Movement propose des mouvements au joueur tant que la variable keepPlaying est égal à 1, le joueur choisi ses mouvements et la fonction lui déplace les cases selon son choix, tout en évitant de sortir de la grille en limitant le déplacement de la case vide entre la 3éme colonne et la 1ere, la 3eme ligne et la 1ere avec une condition if (currentEmptyCompartment.line != 3) et (currentEmptyCompartment.line != 0).
La fonction refuse les mouvement impossible grâce à notre tableau backGrid qui distingue les différents types de case.
Par exemple si on veut déplacer la case vide à la place d'une case de type 1*2 verticalement, la case vide va se déplacer mais pas en une seule ligne mais deux et ainsi de suite.
Si le joueur veut déplacer à gauche la case vide étant donnée que à gauche il y a une case de type 1*2 verticalement le jeu refuse ce mouvement, et pareil pour le cas où il veut la déplacer à droite.
Le joueur peut arrêter de jouer à n'importe quel moment en tapant e ou E ce qui initialise keepPlaying à 0 et fait un break pour sortir de la boucle while.


Fichier Édition Affichage Rechercher Terminal Aide
BIENVENUE SUR QUEL TAQUIN !
| 1 2 3 3 |
| 4 4 5 6 |
| 7 8 9 10 |
| 7 11 12 |
Je vais te mélanger cette grille et tu devra la remettre comme elle était.
Les commandes sont les suivantes :
° Si tu veux déplacer la case vide en bas : appuie sur B.
° Si tu veux déplacer la case vide en haut : appuie sur H.
° Si tu veux déplacer la case vide à droite: appuyerie sur D.
° Si tu veux déplacer la case vide à gauche: appuie sur G.
Si tu veux abandonner, tu peux appuyer sur E pour sortir.
Bon courage !
| 7 1 3 3 |
| 7 4 4 5 |
| 8 10 9 |
| 11 2 12 6 |
À toi de jouer :
PS : Pour gagner, je te propose d'appuyer sur B


PS : Pour gagner, je te propose d'appuyer sur B
b
| 1 2 3 3 |
| 4 4 5 6 |
| 7 8 9 10 |
| 7 11 12 |
Bravo, tu as réussi au bout de 22 mouvement !

Conclusion

A la fin de notre travail on a regardé le cahier des charges pour vérifier si on a pu faire toutes les étapes qu'on a planifier, mais ce n'est malheureusement pas le cas vu qu'on a mal compris l'énoncé du jeu qui demandait de proposer des mouvements au joueur pour gagner tout en suivant la nouvelle situation.
Notre programme propose une seule solution au joueur et s'il ne la suit pas il continu de jouer à sa façon.

L'une des étapes qui nous a pris le plus de temps est de traité le cas des pièces de taille 1x2, l'idée de faire une fonction initGrid pour créer un tableau qui nous aide à distinguer ses cases des autres, était la solution la plus logique pour nous.

La réalisation de ce projet nous a permis de découvrir largement le langage C, sa complexité et aussi de savoir quel sont nos lacunes, points faibles ainsi que les points forts de chacun de nous.

Références

https://openclassrooms.com/fr/
http://www.ai.univ-paris8.fr/~jj/Cours/C/C2019.html
http://www.fil.univ-lille1.fr/~L1S2API/CoursTP/prj_taquin.html