2048

Documentation Technique

MEYER Franck & PISSA Henri

SOMMAIRE

I. Introduction
II. Listes des Fonctions développées
III. Listes des Fonctions utilisées non développées
IV. Tests Unitaires

I. Introduction

Le projet 2048 consiste en la création d'un jeu préexistant, le 2048. Le cahier des charges insiste sur le respect des règles du jeu. Afin de simplifier le développement du jeu, l'intégralité du jeu est décomposée en partie. Chaque partie correspond à une fonctionnalité du jeu regroupant un ensemble de fonctions. La partie I est en charge des prérequis pour créer une partie et des fonctionnalités de base pour faire fonctionner celle-ci.
La partie II est en charge de la gestion du fonctionnement d'une partie.
La partie III est en charge du mouvement au sein de la grille et fait la jonction avec l'interaction offerte à l'utilisateur.
La partie IV est en charge des fonctionnalités permises à l'utilisateur.

II. Liste des Fonctions

a. Partie 1

Structure de données **jeu** :

La structure jeu contiendra tous les éléments nécessaire au fonctionnement d'une partie. Il contiendra les éléments suivants :

Int n	Taille de la grille (initialisé à 4)
Int valMax	Valeur à atteindre pour gagner la partie (initialisé à 2048)
Int nbCasesLibres	Nombre de cases libres sur la grille (initialisé à 0)
Int *grille	Stocke l'adresse de la grille
Int score	Contient le score de la partie en cours (initialisé à 0)

Fonction **initialiseJeu** :

La fonction initialiseJeu permet la création d'une partie en initialisant une structure jeu. Elle permettra notamment à l'utilisateur de pouvoir personnaliser sa partie en modifiant la taille de la grille et la valeur à atteindre pour gagner. Elle sera nécessaire à toute création de nouvelle partie.

Paramètres	
jeu *p	Pointeur vers la structure jeu à initialiser
Int n	La taille de la grille
Int valMax	Score à atteindre pour gagner

Fonction **libereMemoire** :

Permet de libérer la mémoire allouée dynamiquement stockant la grille (malloc). Cette fonction sera utilisée lorsque le joueur quitte le jeu ou lorsqu'il souhaite commencer une nouvelle partie.

Jeu *p	Pointeur vers la structure dont on souhaite effacer la grille

Modifie

Jeu (*p).grille	Libère la grille

Fonction **indiceValide** :

La fonction permet de savoir si l'indice sélectionné existe et correspond à une case de la grille du jeu. Ainsi elle servira de garde-fou, s'assurant constamment que les cases restent dans l'espace mémoire alloué au tableau.

Jeu *p	Pointeur vers la structure jeu contenant les éléments d'une partie
Int i	Le numéro de la ligne
Int j	Le numéro de la colonne

Retourne

1	L'indice existe et est valide
0	L'indice n'est pas valide

Fonction **getVal** :

La fonction permet de récupérer la valeur d'une case valide. La fonction permet à partir de deux indices de récupérer une valeur au sein de d'un tableau à 1 dimension.

Jeu *p	Pointeur vers la structure jeu contenant la partie
Int ligne	Le numéro de la ligne
Int colonne	Le numéro de la colonne

Retourne :

-1	L'indice n'est pas valide
La valeur de la case	L'indice est valide

Fonction dépendante :

- IndiceValide

Cahier des Charges/Test :

- Contrôler la validité de l'indice (indiceValide)
- Retourne la valeur du bon indice

Fonction setVal :

La fonction permet de modifier la valeur d'une case valide.

Paramètre

Jeu *p	Pointeur vers la structure jeu contenant la partie
Int ligne	Numéro de la ligne
Int colonne	Numéro de la colonne
Int val	Valeur remplaçant la valeur existante

Retourne

0	Valeur non changée

1	Valeur changée

Fonction dépendante :

- IndiceValide

Cahier des Charges/Test :

- Controler la validité de la case
- Modifier la bonne case avec la bonne valeur

Fonction **Affichage** :

La fonction assure l'affichage de la partie à l'utilisateur. L'affichage prend la forme d'un tableau de valeurs de n lignes par n colonnes, (n correspondant à la taille de la grille).

- . si la valeur de la case est égale à 0
- La valeur de la case dans tous les autres cas.

L'affichage gère aussi la couleur, affichant une couleur d'arrière-plan à la valeur en fonction de la valeur de la case.

- Pas de couleur d'arrière-plan si la valeur de la case est 0
- Valeur de la case égale à 2 ou à 4 : couleur blanche
- Valeur de la case égale à 8 ou à 16 : couleur bleu
- Valeur de la case égale à 32 ou à 64 : couleur cyan
- Valeur de la case égale à 128 ou à 256 : couleur verte
- Valeur de la case égale à 512 : couleur jaune
- Valeur de la case égale à 1024 : couleur magenta
- Valeur de la case égale à 2048 ou plus : couleur rouge

Paramètre :

Jeu *p	Pointeur vers la structure jeu contenant la partie

Fonction dépendante :

- getVal
- couleurTerminal.c

Cahier des charges/Test :

- Contrôler la valeur récupérée pour chaque case
- Afficher correctement le tableaux (alignement, retour à la ligne), en fonction de la taille de grille
- Afficher la couleur correspondante à la valeur de la case

b. Partie 2

Fonction **caseVide** :

La fonction permet de contrôler si une case est vide (valeur égale à 0).

Paramètres :

Jeu *p	Pointeur vers la structure jeu contenant la partie.
Int i	Numéro de ligne de la case
Int j	Numéro de colonne de la case

Retourne :

0	La case n'est pas vide ou n'existe pas
1	La case est vide (=0)

Fonction dépendante :

- getVal

Cahier des Charges/Test :

- Contrôler si l'indice existe
- Contrôler si la valeur de l'indice récupéré est le bon

Fonction **ajouteValAlea** :

La fonction permet de gérer l'apparition de valeurs aléatoire (2 ou 4) au sein de la grille (case aléatoire vide).

Paramètres :

Jeu *p	Pointeur vers la structure jeu contenant la partie.

Modifie

Jeu (*p).grille	Libère la grille

Fonction dépendante

- caseVide

Cahier des Charges/Test :

- Contrôler si la case est vide.

Fonction **gagne** :

La fonction permet de contrôler si le joueur a gagné sa partie.

Paramètre :

Jeu *p	Pointeur vers la structure jeu contenant la partie.

Retourne :

1	La partie est gagnée
0	La partie n'est pas gagnée

Fonction dépendante :

- getVal

Cahier des Charges/Test :

- Vérifier si dans la grille de jeu une valeur dépasse la valeur définie dans la structure jeu comme la valeur maximale(valMax)

Fonction perdu :

La fonction permet de contrôler si la partie est perdue

Paramètre :

Jeu *p	Pointeur vers la structure jeu contenant la partie.

Retourne :

1	La partie est perdue
0	La partie n'est pas perdue, elle continue

Fonction dépendante :

- getVal

Cahier des Charges/Test :

- Vérifier que la structure ne contient plus de cases libres, grâce à l'entier nbCasesLibres contenue dans la structure jeu.
- Vérifier que la grille n'a pas deux valeurs égales consécutives(indice de la colonne +1, indice de la ligne +n)

c. Partie 3

La partie 3 est la partie gérant principalement le mouvement des cases, ce mouvement a été décomposé en mouvement concernant la ligne ou la colonne. Cela permet de simplifier le programme en ne s'intéressant qu'à un mouvement à la fois. Mais surtout correspond plus précisément au choix de déplacement sur la grille offert au joueur, rendant plus aisé la jonction avec l'interaction du joueur.

Fonction **mouvementLigne** :
Se trouve dans : mouvement.c

La fonction permet de déplacer les valeurs des cases d'une ligne suivant la direction choisie (gauche ou droite). La valeur d'une case est déplacée si la case située à l'indice suivant, (en fonction de la direction), est égale à 0. Si 2 cases juxtaposées sont de mêmes valeurs, alors elles sont fusionnées par addition. Dans tous les autres cas, il n'y a aucun déplacement.

Paramètres :

Jeu *p	Pointeur vers la structure jeu contenant la partie.
Int ligne	Numéro de ligne affecté par le déplacement
Int direction	Direction du déplacement (droite ou gauche)

Retourne

0	Aucune case n'a été déplacé
1	Au moins une case a été déplacée

Modifie

*p.grille[ligne]	Modifie la grille de la structure de jeu.

Cahier des Charges/Test :

- Vérifier que la fonction tasser correctement les espaces entre les valeurs des cases et les 0.
- Vérifier que la fonction additionne 2 cases cote à cote si elles ont même valeurs en fonction de la direction du mouvement.

Fonction **mouvementLignes** :
Se trouve dans : mouvement.c

La fonction permettra de répéter la fonction mouvementLigne , selon la direction du mouvement, pour chaque lignes de la grille et ceux qu'elle que soit la taille de la grille, assurant une certaine adaptabilité. Elle évite et simplifie le phénomène de boucle mis en jeu.

Paramètres :

Jeu *p	Pointeur vers la structure jeu contenant la partie.
Int direction	Direction du déplacement (droite ou gauche)

Retourne :

0	Aucune case n'a été déplacée
1	Au moins une case a été déplacée

Modifie :

*p.grille	Modifie la grille de la structure de jeu.

Fonction dépendante :

- mouvementLigne

Cahier des Charges/Test :

- Vérifier que la fonction mouvementLigne impacte toute les lignes de la grille.

Fonction **mouvementColonne** :
Se trouve dans : mouvement.c

La fonction permet de déplacer les valeurs des cases d'une colonne suivant la direction choisie (gauche ou droite). La valeur d'une case est déplacée si la case située à l'indice suivant est égale à 0. Si 2 cases juxtaposées sont de mêmes valeurs, alors elles sont fusionnées par addition. Dans tous les autres cas, il n'y a aucun déplacement.

Paramètres :

Jeu *p	Pointeur vers la structure jeu contenant la partie.
Int ligne	Numéro de ligne affecté par le déplacement
Int direction	Direction du déplacement (haut ou bas)

Retourne :

0	Aucune case n'a été déplacée
1	Au moins une case a été déplacée

Modifie :

*p.grille	Modifie la grille de la structure de jeu.

Cahier des Charges/Test :

- Vérifier que la fonction tasser correctement les espaces entre les valeurs des cases et les 0.
- Vérifier que la fonction additionne 2 cases cote à cote si elles ont même valeurs en fonction de la direction du mouvement.

Fonction **mouvementColonnes** :
Se trouve dans : mouvement.c

La fonction permettra de répéter la fonction mouvementColonne , selon la direction du mouvement, pour chaque colonnes de la grille et ceux qu'elle que soit la taille de la grille, assurant une certaine adaptabilité. Elle évite et simplifie le phénomène de boucle mis en jeu.

Paramètres :

Jeu *p	Pointeur vers la structure jeu contenant la partie.
Int direction	Direction du déplacement (haut ou bas)

Retourne :

0	Aucune case n'a été déplacée
1	Au moins une case a été déplacée

Modifie :

*p.grille	Modifie la grille de la structure de jeu.

Fonction dépendante :

- mouvementColonne

Cahier des Charges/Test :

- Vérifier que la fonction mouvementColonne impacte toute les colonnes de la grille.

Fonction **mouvement** :
Se trouve dans : mouvement.c

La fonction va assurer la jonction entre la saisie de l'utilisateur et le mouvement appliqué à la grille, interprétant chaque commande de l'utilisateur.

Paramètres :

Jeu *p	Pointeur vers la structure jeu contenant la partie.
Int direction	Direction du déplacement (droite ou gauche)

Retourne :

0	Aucune case n'a été déplacée
1	Au moins une case a été déplacée

Modifie :

*p.grille	Modifie la grille de la structure de jeu.

Fonction dépendante :

- mouvementColonnes
- mouvementLignes

Cahier des Charges/Test :

- Vérifier que le bon mouvement a été effectué.

Fonction **saisieMD** :
Se trouve dans : saisieMD.c

La fonction gère les inputs de l'utilisateur à l'aide des quatres flèches directionnelles et de la touche échap.

Retourne :

-1	Touche Echap saisie // Retourner au menu
0	Flèche Bas saisie // Descendre les éléments de la grille
1	Flèche Droite saisie // Déplacer vers la droite les éléments de la grille
2	Flèche Haut saisie // Monter les éléments de la grille
3	Flèche Gauche saisie // Déplacer vers la gauche les éléments de la grille

Fonction dépendante :

- saisieM
- couleurs_terminal

Cahier des Charges/Test :

- Vérifier que la valeur retourné correspond au mouvement effectué.

d. Partie 4

Fonction **sauvegarde** :
Se trouve dans : sauvegardes.c

La fonction permet d'enregistrer la structure d'un jeu dans un fichier binaire.
On enregistre indépendamment chaque élément de la structure.
Pour gérer la sauvegarde multiple, il faut juste créer un fichier par sauvegarde. On limite ici la sauvegarde à 3. Pour savoir sur quel fichier enregistrer nous laissons le joueur choisir son emplacement. La précédente sauvegarde sera écrasée.

Paramètres :

Jeu *p	Pointeur vers la structure jeu contenant la partie.
Int emplacement	Choix de l'emplacement de la sauvegarde (5 choix possibles = 5 sauvegardes disponibles au maximum)

Retourne :

0	Problème de sauvegarde
1	La sauvegarde s'est déroulée correctement

Cahier des Charges/Test :

- Vérifier que lors de la sauvegarde il n'y a aucune erreur d'enregistrement (copie des données)

Fonction **chargement** :
Se trouve dans : sauvegardes.c

La fonction permet de charger la structure d'un jeu contenu dans un fichier binaire vers la partie en cours (initialisé à ce moment).
On charge indépendamment chaque élément de la structure.
Pour savoir quel fichier charger nous laissons le joueur choisir son emplacement grâce à la au paramètre emplacement. La partie en cours est perdue.

Paramètres :

Jeu *p	Pointeur vers la structure jeu contenant la partie.
Int emplacement	Choix de l'emplacement de la sauvegarde (5 choix possibles = 5 sauvegardes disponibles au maximum)

Retourne :

0	Problème de sauvegarde

1	La sauvegarde s'est déroulée correctement

Cahier des Charges/Test :

- Vérifier que lors de le chargement a récupéré la bonne sauvegarde
- Vérifier que la sauvegarde n'est pas corrompue

Fonction **menu** :
Se trouve dans : menu.c

La fonction menu est en charge de l'interface graphique entre l'utilisateur et le programme. Le menu offre des choix avec lesquels l'utilisateur peut interagir grâce à la fonction saisieMD(). Tant qu'il n'a pas choisie une option valide, la fonction boucle sur le menu.

Cahier des Charges/Test :

- Vérifier que le choix dans le menu fait par l'utilisateur est le bon.

III. Listes des Fonctions utilisées non développées

- couleurs_terminal.c
  La fonction contenue dans le fichier gère l'affichage de la couleur sur le terminal. Permettant notamment de changer la couleur, de la police d'un texte mais aussi la couleur d'arrière-plan.
  Pour s'en servir il faudra remplacer le printf normal et ajouter 2 nouveau paramètres.
  color_printf(couleur avant-plan,couleur arrière-plan, texte)
- saisieM.c
  La fonction contenue dans le fichier permet de capturer un input sur l'une des touches fléchées ou sur la touche Echap.

IV. Test Unitaires

Les Tests Unitaires sont regroupés dans la dossier Test_Unitaire. Les tests sont décomposés par partie (par répertoire). Un fichier test est composé d'une batterie de test pour chaque fonction d'une partie.
Pour pouvoir exécuter un le test d'une partie, un Makefile a été créé, il suffit donc simplement d'exécuter la commande make dans le répertoire du test.
Le test, une fois exécuté, va afficher le résultat de tous les tests de la partie ont été passés avec succès et sinon le test échoué.