Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes - RABAT

Rapport de Projet de programmation : Othello

Réalisé par :

Mohamed Zahar

Encadré par :

Pr. Mahmoud EL Hamlaoui

Année Scolaire 2020/2021

Remerciements :

Tout d'abord, je veux exprimer mes sincère gratitudes pour la confiance,le support et l'importance que m'a accordé professeur EL Hamlaoui Mahmoud,je le remercie pour cette opportunité qui m'a permis de developper et polir mes compétences dans la programmation en langage c, cela m'a sûrement donner un aperçu de la vie professionnelle qui m'attend.

Mes plus grands remerciements vont aussi aux professeurs dont les cours m'ont guidé dans mon travail citons parmis eux : Mr. Abdelatif EL Faker,Mr. Ahmed Ettalbi,et Mr. Hatim Guermah.

Table des matières

Chapitre 1

Présentation du projet

Introduction

Le projet c s'agit de produire un programme d'environ 500 lignes de code afin de valider les compétences des cours :« Algorithmique », « Technique de programmation » et « Structures de données ». Le programme correspond à20 heures de travail effectives en langage C. Les étudiants travaillent en binôme et bénéficient des conseils d'unprofesseur encadrant.

1.1 Sujet

Le jeu de Othello est un jeu combinatoire abstrait, sans hasard, avec information complète et parfaite. Deux joueurs, "noir" et "blanc" s'affrontent. Le jeu se joue sur un plateau de 64 cases (8x8) ; le jeu de déroule à tour de rôle où un joueur doit déplacer son disque,si un joueur ne peut plus se déplacer alors il céde son tour à son adversaire.le jeu se termine quand les deux joueurs ne pevent plus se déplacer et le joueur qui le plus grand nombre de disques gagne :

FIGURE 1.1 – Planche du du Jeu Othello

0. Notice du projet de programmation 2020/2021

1.2 Étymologie

Le mot "Othello" fait référence à la pièce de théatre de Shakespeare "Othello,le Maure de Venise",le jeu parle du conflit entre deux personnes de la pièce Othello et Desdemona, la couleur noire se réfère à Desdemona parce qu'elle était noire, et la couleur blanche qui se réfère à Othello qui était blanc, évidemment l'une d'elles gagnait le plus après les deux personnes un drame instable.

la couleur verte du plateau est une représentation de la bataille menée par le général Othello dans les champs, et les boules qui deviennent blanches sont les ennemis qui sont devenus sa possession pendant la bataille.. [1]

FIGURE 1.2 – Othello and Desdemona

1.3 Principe du jeu

tout comme le jeu Reversi, les deux joueurs commencent par placer deux disques chacun au centre du plateau, les disques doivent être placés de manière à pouvoir se consommer, ce qui signifie que chaque disque peut être consommé par les deux autres disques adverses, et puis un à un va commencer à déplacer leurs disques, la seule façon dont un mouvement est possible est quand il consomme un disque adversaire ou plus, la position choisie est accessible par plusieurs disques puis tout le disque adversaire entre la position choisie et le disque en mouvement va à consommer.

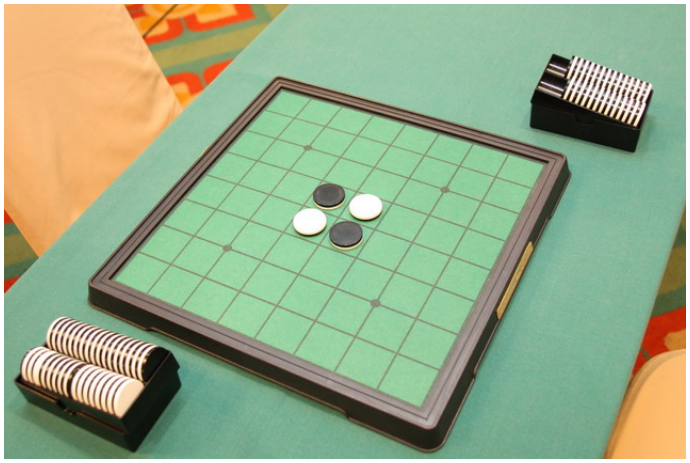

FIGURE 1.3 – La table Othello

Chapitre 2

Analyse théorique

2.1 Cahier de charges

Le cahier de charge permettra d'élaborer les instructions et indications qui mennent à la réalisation etl'implémentation du code,ce code aura pour but de repondre aux points suivants :

2.1.1 Conception de la table

Le jeu commencera avec 4 disques placés dans le milieu de la table, de la manière que chaque disque est enface avec son adversaire, alors que les autres positions vide sont replises avec des points ".",la table commence de 1 et 8 et chaque position est indéxée avec sa ligne et sa colonne.

	1	2	3	4	5	6	7	8
1	.	.	.	.	.	.	.	.
2	.	.	.	.	.	.	.	.
3	.	.	.	.	.	.	.	.
4	.	.	.	W	B	.	.	.
5	.	.	.	B	W	.	.	.
6	.	.	.	.	.	.	.	.
7	.	.	.	.	.	.	.	.
8	.	.	.	.	.	.	.	.

FIGURE 2.1 – Exemple proposé de la table

2.1.2 But du jeu

Le but de chaque joueur dans le jeu est de consommer plus de disque de que son adversaire,dans son mouvement d'une case à une autre,le joueur qui le nombre de disque le plus grand sera le vainqueur.

La condition d'arrêt :Le jeu termine qu'on toutes le cases sont remplises ou quand les deux joeurs n'ont plus de mouvements possibles.

2.1.3 Fonctionnement du jeu

un tour se fait de la maniére suivante :
— Le joueur choisit une ligne et une colonne où il veut se deplacer.
— puis le joueur part vers cette postion en consommant les diques de son adversaire.cela s'appliquera sur touts ses disques qui peuvent partie vers la position choisie ;
— puis l'adversaire joue à tour de rôle. ;

— si un joueur n'a plus de deplacement l'autre joueur doit jouer jusqu'il n'aura plus de place possible,et donc le jeu terminera.
— le joueur qui a le meilleur score est celui qui a pu limiter les mouvement de son adversaire le plus tôt possible.

2.1.4 Conception

on doit venir à porogrammer un jeu qui **contrôle la sécurité**, et donne l'option au joueur de **choisir entre charger une partie déja sauvegarder**, de **jouer une nouvelle partie**,ou d'**afficher les dix meilleurs scores**.

Si le joueur choisis de jouer une nouvelle partie, il lui demande son nom ,ensuite s'il veut jouer contre un robot ou un être humain,s'il choisit de jouer contre un être humain il lui demande alors le nom du deuxieme joueur, aprés le choix des nom le programme demandera au premier joueur de choisir un avatar "b" ou "w", et affecte automatiquement l'opposé à son adversaire.le jeu ensuite se fait à tour de rôle et à chaque fois qu'un joueur fait un mouvement il demande au suivant s'il veut recommencer et sauvegarder la partie ou continuer de jouer.Si le joueur demande l'affichage des dix meilleurs scores, le programme lit un fichier où les vainqueur avec leur score sont enregistrer et afficher les dix gagnant qui ont le score le plus bas.Si le joueur choisit de charger une partie, le programme lui demandera s'il veut jouer contre un robot ou un être humain,comme dans le cas de jouer une nouvelle partie,mais dans ce cas il ne demandera pas aux joueur de choisir l'avatar.le jeu doit permettre ausssi d'afficher l'historique des mouvements à la fin et durant le jeu.

2.2 Résolution

On discutera dans cette partie la méthode de résolution des contraintes

2.2.1 Conception de la planche

On commence par créer la table initiale,puis on passe pour donner les places possibles pour chaque joueur,on commence par la transformation de la table de la manière suivante,on encadre les disques initiales,en remplacant ce cadre par des 1 et toutes autres positions par des 0,cela se fait dans un tableau d'entiers,ensuite on ne laisse que les position accessible à chaque joueur c'est à dire les positions dans les deux diagonales,la ligne,et la colonn, en remplacant ces positions par des 2,puis on doit éleminer les places occupées par des "w" et des "b" en les remplacant par des 0.puis on choisit que les position où on a des 2 et on étudie les cas où le passage à 2 consomme un disque adversaire ou non.

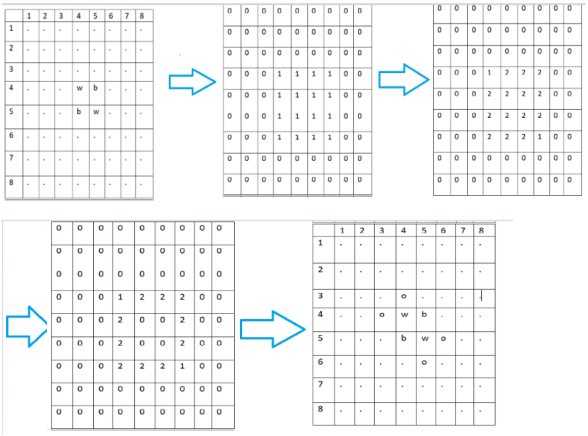

FIGURE 2.2 – Exemple de procedure de cretion de table pour un joueur 'b'

2.2.2 Condition d'arrêt

Pour trouver la condition d'arrêt on passe par les étapes suivantes :

— Etape 1 :on génére la table des positions possible pour le premier joueur.

— Etape 2 :on compte le nombre des 'o' dans la table .

— Etape 3 :on génére la table des positions possible pour le deuxième joueur.

— Etape 4 :on compte le nombre des 'o' dans la table .

— Etape 5 :on compare les deux nombres générés s'ils sont égaux à 0 le jeu se termine sinon on continue.

2.2.3 Recommencer et sauvegarder une partie

Le premier joueur jou son tour et on demande au joueur suivant s'il veut recommencer sauvegarder la partie,s'il veut recommencer la table courante sera enregistrée dans un fichier donr il choisit le nom,et la partie recommence avec un tableau initial où le deuxième joueur est celui qui commence.

NB :Le fichier sera ouvert sous le mode 'a+'

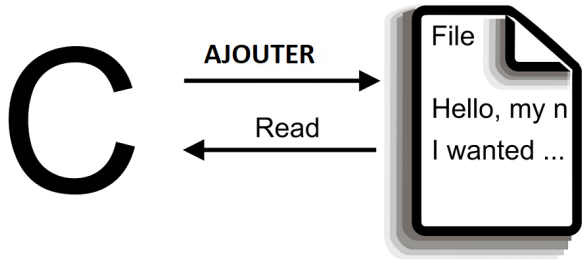

FIGURE 2.3 – lecture et ajout dans un fichier c

2.2.4 Affichage des dix meilleurs scores

Aprés la fin de chaque partie on enregistre les caractéristiques du vanqueur dans un fichier c on utilisant le mode 'a+',et puis on prend les scores des joueurs dans le fichier et on les tries par ordre croissant,puis on affiche les 10 premiers scores dans l'ordre croissant avec leur nom.

2.2.5 l'enregistrement des joueurs et leurs caractéristiques

Cela se fait à la fin du jeu,comme dans le cas de l'affichage des dix meilleurs scores,sauf qu'ici on enregistre les caractéristiques des deux joueurs.

2.2.6 Jouer contre un non humain

Dans cette partie du jeu, le joueur choisi de jouer contre un non humain,pour en faire,le programme cherche les positions possible de l'adversaire et choisi une arbitrairement.

2.2.7 Authentification

On doit demander à joueur de taper son username et puis si cela n'existe pas on lui donne l'option doit de créer un nouveau code ou de quitter,sinon on lui demande de taper son mot de passe si cela est faux,le joueur est donner l'option de créer un nouveau compte ou de quitter.

Chapitre 3

Réalisation du projet

Introduction

Dans cette parie on va passer à la réalisation du projet en citons les problèmes rencontrés durant la programmation du jeu et et la démarche de sa réalisation :

3.1 Problèmes rencontrés

3.1.1 Problèmes de divisions de travailles

j'ai rencontré des problèmes concernant le travaille en groupe,il y'avait des malentendus et des incompréhension qui étaient difficiles à résoudre.

3.1.2 Problèmes de compréhension

En jouant le jeu la première fois je n'ai pas pu remarqué plusieurs détails comme le fait que quand un joueur décide de se déplacer vers une positions,tous les disques qui mennet à cette position vont de déplacer.

3.1.3 Problèmes d'organisation du code

Vu que la programmation du jeu est sépqrée au jeu j'au eu des problèmes dans la rédactions des fonction,car à chaque fois que je passe d'un point à un autre des modifications doivent être faites sur les fonctions.

3.1.4 Algorithme MiniMax Alpha Beta

J'ai compris le principe du code mais j'ai eu des probléme dans la conception,je n'ai pas comprendre ce que je dois mettre dans les feuilles ainsi que comment je peux changer la stucture de l'arbre aprés chaque mouvement.

3.1.5 Interface Graphique

Je n'ai pas arrivé à réaliser l'interphace graphique à cause des problèmes de gestion du temps.

3.2 Signification des fonctions du code

3.2.1 Structure de données utilisée

On utilise la structure "joueur" qui contient le nom,le score,et le logo.

3.2.2 Créer table

la fonction creer_table() permet de créer la table initiale de caractère, avec deux disques blanc et deux disques noires placés dans le centre du tabeau.

3.2.3 Donner les places possibles d'un joueur

la fonction encadrant(char **table) permet d'encadrer les disque des joueurs comme vues dans la deuxième image de la figure 2.2 .
la fonction places_possible(char player,char **table,int **table1) prend en argument le joueur c'est à dire 'w' ou 'b',la table de caractères,et la table encadrant pour enfin donner la deuxième table de la figure 2.2.
la fonction place_libre(int **table1,char **table) prend en argument la table des encadrants et la table de caractères pour ensuite donner l'image 3 de la figure 2.2.

3.2.4 Donner l'adversaire d'un joueur

La fonction adversaire(char player) prend un argument 'w' ou 'b' et renvoie son adversaire.

3.2.5 Faire un déplacement

3.2.5.1 Les fonction check

les fonctions suivant vont chercher si une position peut être accessible par le joueur c'est à dire elle va chercher dans les positions qui précédent la position qui précéde encore la position desirée, et si elle trouve le joueur donc elle retourne un 1 sinon un 0 :

- La fonction check_in_diag_sup(int i,int j,char player,char **table)cherche dans la diagonale mais en bas.
- La fonction check_in_diag_inf(int i,int j,char player,char **table) cherche dans la diagonale en haut.
- La fonction check_in_ligne_droite(int i,int j,char player,char **table) cherche dans la line a droite.
- La fonction check_in_ligne_gauche(int i,int j,char player,char **table) cherche dans la ligne a gauche.
- La fonction check_in_colonne_up(int i,int j,char player,char **table) cherche dans la colonne en haut.
- La fonction check_in_colonne_down(int i,int j,char player,char **table) cherche dans la colonne en bas.
- La fonction check_in_reversed_diag_gauche(int i,int j,char player,char **table) cherche dans la deuxième diagonale en bas.
- La fonction check_in_reversed_diag_droite(int i,int j,char player,char **table) cherche dans la deuxième diagonale en haut.
- La fonction check_in_diag_inf(int i,int j,char player,char **table) cherche dans la diagonale mais en haut.

3.2.5.2 La fonction de déplacement

La fonction deplacement(char **table, char player) suivante va permettre de donner les positions ou le joueur peut se déplacer c'est à dire les position ou son deplacement va permettre de consommer son adversaire ; en fait les conditons de la fonction permettent de verifier si une des positions qui encadrent la position desirée est occupée par son adversaire puis elle cherche s'il existe un disque du joueur qui va consommer le disque de l'adversaire pour passer à la position desirée,si cela est verifié donc on donne à cette postion la valeur 'o',en utilisant l'exemple precedent et en choisissant de jouer avec 'b' la fonction suivante va nous rendre le dernier tableau de la figure 2.2.

3.2.5.3 Les fonctions check1

Les fonctionts suivantes vont nous permettre de calculer le nombre de disques qu'un joueur doit consommer dans son mouvement,elles vont donc calculer le nombre des disques adversaires qui éxistent avant la premiére apparition d'un disque du joueur.

- La fonction check1_in_diag_sup(int i,int j,char player,char **table)cherche dans la diagonale mais en bas.
- La fonction check1_in_diag_inf(int i,int j,char player,char **table) cherche dans la diagonale en haut.
- La fonction check1_in_ligne_droite(int i,int j,char player,char **table) cherche dans la line a droite.
- La fonction check1_in_ligne_gauche(int i,int j,char player,char **table) cherche dans la ligne a gauche.
- La fonction check1_in_colonne_up(int i,int j,char player,char **table) cherche dans la colonne en haut.
- La fonction check1_in_colonne_down(int i,int j,char player,char **table) cherche dans la colonne en bas.
- La fonction check1_in_reversed_diag_gauche(int i,int j,char player,char **table) cherche dans la deuxième diagonale en bas.

— La fonction check1_in_reversed_diag_droite(int i,int j,char player,char **table) cherche dans la deuxième diagonale en haut.
— La fonction check1_in_diag_inf(int i,int j,char player,char **table) cherche dans la diagonale mais en haut.

3.2.5.4 Les fonctions écraser

Les fonctions qui suivent vont permettre le mouvement d'un joueur à une position en consommant une boule ou plus du joueur précédent.
— La fonction ecraser_diagonal_sup(char **table,int i,int j,char player) permet d'écraser les disque adversaire dans la digonale en bas.
— La fonction ecraser_diagonal_inf(char **table,int i,int j,char player) permet d'écraser les disque adversaire dans la digonale en haut.
— La fonction ecraser_ligne_gauche(char **table,int i,int j,char player) permet d'écraser les disque adversaire dans la ligne à gauche.
— La fonction ecraser_ligne_droite(char **table,int i,int j,char player) permet d'écraser les disque adversaire dans la ligne à droite.
— La fonction ecraser_colonne_up(char **table,int i,int j,char player) permet d'écraser les disque adversaire dans la colonne en haut.
— La fonction ecraser_colonne_down(char **table,int i,int j,char player) permet d'écraser les disque adversaire dans la colonne en bas.
— La fonction ecraser_reversed_diagonal_gauche(char **table,int i,int j,char player) permet d'écraser les disque adversaire dans la deuxième diagonale en bas.
— La fonction ecraser_reversed_diagonal_droite(char **table,int i,int j,char player) permet d'écraser les disque adversaire dans la deuxième diagonale en haut.

3.2.5.5 La fonction MOVE

La fonction suivante demande du joueur la position où il veut aller et elle vérifie si c'est une position sccessible c'est a dire s'il existe un 'o' dans cette place,si cela est vérifier donc elle écrase les disques de l'adversaire et remplace le 'o' par le disque du joueur sinon elle retourne le message "deplacement illegale* et elle stock les valeurs de ces mouvements dans un fichier "mouvement.txt" puis elle demande au suivant s'il veut recommencer ou non ,s'il choisit de recommencer donc elle sauvegarde le dernier tableau dans un fichier dont le joueur choisi le nom puis elle recommence le jeu en commencant avec le joueur qui a choisit de recommencer.

NB : dans le cas d'un robot avec un être humain la fonction ne demande qu'à l'être humain s'il veut recommencer.

3.2.6 Les fonctions count

le comptage des 'o'

La fonction count_o(char **table) donne le nombre des places possibles c'est à dire le nombre des 'o'.

le comptage des 'b'

la fonction count_b(char **table) calcule le nombre des 'b' dans une table prise en argumant.

le comptage des 'w'

la fonction count_w(char **table) calcule le nombre des 'w' dans une table prise en argumant.

3.2.7 Affichage de l'historique des mouvements

La fontion historique_mouvements(char file[]) permet d'afficher l'historique des mouvement à partir d'un fichier "mouvement.txt" en ajouttant les données du fichier à un tableau de chaine de caractères exemple :T[0] est égale a "b=(2,1)" puis elle affiche le contenue du tableau.

3.2.8 Affichage de la table de caractères

La fonction afficher_table(char **table) permet d'afficher la table de caractères.

3.2.9 Sauvergarde des caractéristiques des joueurs

La fonction sauvegarder(joueur player1,joueur player2) permet de sauvegarder les caracteristiques des deux joueurs dans un fichier "players.txt",elle prend en argument deux variables de type joueur.

3.2.10 jouer à tour de rôle

La fonction play_round(char player,char joueur1[],char joueur2[],char **table,int s,int d) va permettre de jouer le jeu à tour de role de la manière qu'elle affiche le nom du joueur , l'historique des mouvement et le score après chaque mouvement,et après la fin du jeu elle enregistre le joueur gagnant dans un fichier "gagnants.txt" et elle enregistre les caractéristiques des deux joueurs dans un fichier "players.txt".

3.2.11 Jouer une nouvelle partie

La fonction jouer(char **table,char joueur1[],char joueur2[],int d,int s) permet de jouer le jeu et donne le logo du premier jouer 'w' ou 'b' selon son choix s'il est un humain sinon il lui donne 'b'.

3.2.12 Charger une partie

La fonction charger_partie(char joueur1[],char joueur2[],int s,int d) permet de charger une partie déja sauvegarder de la manière à prendre les données du fichier déja nommé et les ajouter à un tableau de caractères on va donc créer un nouveau tableau qu'on va utiliser pour jouer la partie.

3.2.13 affichage des dix meilleurs scores

La fonction dix_meilleurs_scores(char file[]) permet d'afficher les dix meilleurs scores en utilisant le fichier "gagnants.txt" et en prenant juste le score des gagnants puis on l'ajoute à un tableau d'entiers et on le trie par ordre croissant, puis on cherche dans le fichier le joueur qui a son image dans le tableau trier et on l'affiche , on se limite des 10 meilleurs scores.

3.2.14 Jouer le jeu

La fonction play_game() permettra de demander au joueur de choisir parmis 3 options : de charger une partie, de jouer une nouvelle partie ou d'afficher les dix meilleurs scores s'il choisit de charger une partie alors elle lui demande si 2 humains vont jouer ou bien 2 robots ou bien 1 humain et un robot si un humain va jouer alors elle lui demande de saisir son nom , c'est aussi le cas pour le choix de jouer une nouvelle partie, mais s'il choisit de charger les 10 meilleurs scores donc elle n'affiche que les 10 meilleurs scores.

3.2.15 Sécurité

3.2.15.1 Créer un compte

La fonction creer_compte() au joueur de créer un nouveau compte et prend en compte le cas ou le username éxiste déja,elle enregistre les données dans deux fichiers, le premier ne contient que le username nommé "users.txt",et l'autre contient le username et le mot de passe nommé "identification.txt".

3.2.15.2 Authentification

La fonction authentification() de vérifier l'accés d'un utilisateur qui a déja un compte,si elle arrive à trouver ses informations dans les fichiers donc elle lui donne accés au jeu,sinon elle lui demande s'il veut créer un compte,le joueur aura ensuite le choix entre créer un compte ou quitter le jeu.

3.2.15.3 Manipulation des données

La fonction donnees_user() représente la fonction de gestion de données puisqu'elle donne au joueur le choix entre créer un compte ou se connecter. après l'étape d'authentificaton elle lui accorde l'accés au jeu.

3.2.16 Organisation du code

3.2.16.1 Fichier header.h

Ce fichier contient les fonctions, la déclaration des variables et des structures de données.

3.2.16.2 Fichier functions.c

Fichier de prototyping :un appel aux fonctions utilisées.

3.2.16.3 Fichier main.c

ne contient que la fonction donnees_user() qui permet de jouer le jeu.

Bibliographie

[1] <https://openclassrooms.com/forum/sujet/determiner-la-taille-d-un-fichier>, 2013. [Online ; accessed 26-January-2021].

[2] <https://forums.commentcamarche.net/forum/affich-27337373-masquer-mot-de-passe>, 2013. [Online ; accessed 07-february-2021].

[3] <https://github.com/med-zr/Othello>, 2021. [lien du projet sur github].