LC4 - ABALONE

1. Le plateau

Le plateau de jeu a été représenté à l'aide d'une structure composée de trois variables : un tableau correspondant aux cases du plateau, et deux entiers comptant le nombre de pions blancs et noirs sortis.

Le tableau, crée en taille fixe à l'aide d'un define, est carré. Ainsi, les cases non utilisées pour le jeu (c'est à dire les cases n'entrant pas en compte dans un hexagone) sont représenté par le caractère '0', les pions blancs représenté par 'B', les pions noirs par 'N' et les cases vides par ' . '.

Ce choix a été fait afin de simplifier la création du plateau, ainsi que son utilisation afin d'éviter de devoir allouer dynamiquement la taille de chacune des lignes du tableau.

2. Le jeu

Lors de l'entrée dans le programme, chacun des arguments entrés par l'utilisateur sera parsé modifiant ainsi le jeu en fonction des options.

Les options disponibles :

* -t : Correspond au mode de test.

* -c : Nécessite un argument correspondant à un fichier. Permet de charger une partie.

* -B|-N : Permet de changer le comportement du joueur correspondant.

Lors d'une partie normale (c'est à dire sans aucune option particulière), le jeu se lancera et demandera au premier utilisateur (le joueur Blanc), d'entrer un coup, et ainsi de suite. Si le coup entré n'est pas valide, un message d'erreur s'affichera et demandera à l'utilisateur d'entrer à nouveau un coup.

Afin de tester la validité d'un coup, nous possédons plusieurs fonctions. La première parsera la commande afin de vérifier si elle est conforme à la notation du programme à l'aide d'une regex. Ensuite, nous remplissons un tableau avec pour objectif de lier chaque case de départ sélectionnée avec une case d'arrivée. Ce tableau sera ensuite envoyée dans une autre fonction, vérifiant la validité du mouvement (en fonction des règles du jeu : nombre de billes adverses, nombre de billes sélectionnées...). Et enfin, la toute dernière fonction appelée servira à effectuer le mouvement sur le tableau.

Mais le jeu permet aussi d'utiliser d'autres commandes :

* exit : permet de quitter le programme directement, en libérant toute mémoire en cours d'utilisation.

* undo : permet d'annuler le dernier coup, et seulement le dernier. Pour chacun des coups joués, une sauvegarde de l'ancien plateau est gardé, afin de permettre l'utilisation de cette commande.

* redo : permet de refaire le coup dernière annulé. Il devient impossible après chaque coup joué. Le plateau annulé est sauvegardé dans une variable pour permettre l'utilisation de cette commande.

* save : sauvegarde l'état courant de la partie dans le format standard donné par le projet (permettant donc l'utilisation de l'argument -c, qui sera détaillée plus tard). Un nom de fichier de sera demandé par l'utilisateur, et sera sauvegardé automatiquement dans le dossier « savefile » du projet.

Lorsqu'un des deux joueurs perd 6 pions, le jeu s'arrête et affiche le gagnant.

Tout le jeu est géré par une boucle while infinie, permettant de gérer les exceptions (erreurs dans la commande, autre commande ne comptant pas comme un coup) à l'aide de « continue », et aussi, lors de la fin du jeu, de faire un simple « break » afin de sortir de cette boucle.

3. Les arguments

L'argument -c :

Charge un fichier de sauvegarde permettant de changer l'état du jeu de départ. C'est notamment grâce à cet argument qu'il devient possible de jouer aux différentes variantes du jeu.
Deux fonctions sont utilisées pour permettre le chargement d'un fichier.
La première effectue une première lecture, et permet de vérifier qu'il y a bien le bon nombre de caractère présent de le fichier, et qu'il respecte bien le standard donné par le sujet du projet. Ainsi, au moindre problème, le programme est arrêté permettant d'éviter tout erreur par la suite.
La seconde relis le fichier, et remplis les variables de manière à ressembler au contenus donné par le fichier (joueur suivant ainsi que le tableau...).

L'argument -t :

Permet de mettre en place le mode test. Ce mode servira, à l'aide d'un fichier, à donner une suite de commande au programme affin d'effectuer un test automatique d'une partie.
Le programme s'arrêtera si une commande devient impossible à effectuer.
Pour permettre ce mode, des conditions ont été ajouté afin de supprimé les prompts d'utilisateurs, et affiche seulement le coup joué.

Les arguments -B|-N :

Change l'état du joueur Blanc ou du joueur Noir.
On peut donc choisir si un joueur est humain ou si il est une IA. Dans ce projet, seul l'ia aléatoire à été réalisée...
Les deux mots clés permettant de changer les états : ia, human.