```
#ifndef GRILLE_H
#define GRILLE_H
#include <iostream>

struct jeton
{
        char forme; // forme du jeton tel qui apparaitra lors de l'affichage
        int couleur;// couleur du jeton tel qui apparaitra lors de l'affichage
        bool marquage;// permet de marqué le jeton pour évité la reutilisation des même
        jetons dans deux alignements de même type(diagonale gauche et diagonale droite)
};

class grille
{
    /* Classe permettant la structuration et la manipulation de la grille de jeu.
   *
   * Cette classe definit une grille comprenant ses données. Tout lien avec la grille
   passera par cette classe.
   * Cette classe possede des données la caracterisant et ds méthode pour la manipuler. La
   grille de jeu est comprise
   * dans la classe Partie, c'est donc cette classe qui la dirige. La classe grille ne peut
   donc pas utilisé les méthode
   * des classe Partie et joueur.
   */
        private:
                int nb_ligne; /*modelise le nombre de ligne de la grille de jeu*/
                int nb_colonne; /*modelise le nombre de colonne de la grille de jeu*/
                int nb_jeton; /*modelise le nombre de jeton que les joueurs doivent aligner
                pour faire un alignement*/
                int nb_alignement; /*modelise le nombre d'alignement que le joueur doit
                faire pour gagner*/
                int nb_joueur; /*modelise le nombre de joueur s'affrontant sur cette partie*/
                int *al_J; /*Tableau d'entier contenant le nombre d'alignements de chaque
                joueur */
                int resultat; /*variable qui permet de connaitre le resultat d'une partie
                (partie gagnée par un joueur ou partie nulle)*/
                jeton **Table; /*Tableau à deux dimmension de jeton modelisant la grille de
                jeu en elle-meme */
                jeton *all_jeton; /*Tableau de jeton contenant les jetons de chaque joueur*/
                int *n_jeton; //nombre maximal de jetons aligné par joueur
        public:

                grille();
                /*
                 * Construit une grille vide.
                 */

                grille(const grille& G_tmp);
                /*
                 * Constructeur de recopie.
                 */

                ~grille();
                /*
                 * Detruit une grille.
                 */

                void affichage();
                /*
                 * Affiche la grille de jeu avec les diffrent jetons (avec leur forme et
                 leur couleur).
                 * Cette fonction est appelée par:
                 * - jouer (class Partie)
                 * - victoire (class Partie)
                 */

                void saisie();
                /*
                 * Permet la saisie des données de la grille avant de la préremplir.
                 * Cette fonction est appelée par :
                 * -menu_pricipal (class Partie)
```

```
 */

void preremplir_grille();
/*
 * Permet grace aux donnée saisie de faire une grille vide de x colonne et y
 ligne
 * Cette fonction est appelée par :
 * -Saisie (class grille) après la saisie initiale
 * -tourner_horaire (class grille) en cas de pivot horaire
 * -tourner_antihoraire (class grille) en cas de pivot antihoraire
 */

void enlever_marquage_alignement();
/*
 * Retire tout les marquage de la grille pour faire la vérification en
 diagonale d'un autre type
 * Cette fonction est appelée par :
 * -verification_diagonale_1 (class grille)
 * -verification_diagonale_2 (class grille)
 */

void initialiser_alignement();
/*
 * Retire tout les alignement des joueurs avant chaque nouveau coup car si
 un pivot ou un changement de jeton
 * est effectué, il y a alors perte ou gain d'alignement pour chaque joueur.
 * Cette fonction est appelée par :
 * -tour_du_joueur (class Partie)
 */

void placer(int colonne, jeton le_jeton, int joueur);
/*
 * Place un jeton dans la colonne de la grille choisit par le joueur. Le
 jeton se met alors sur un autre jeton
 * si il y adeja un jeton dans cette colonne ou en bas de la colonne dans le
 cas contraire
 * Cette fonction est appelée par :
 * -jouer (class Partie)
 */

void pivoter(std::string pivot, jeton le_jeton, int joueur);
/*
 * \brief Pivote la grille de jeu
 *
 * Fait pivoter la grille de jeu dans le sens choisit par le joueur grace a
 deux méthode (vers_la_droite et
 * tourner_horaire pour un pivot dans le sens horaire ou vers_la_gauche et
 tourner_antihoraire pour un pivot en
 * sens antihoraire)
 * Cette fonction est appelée par :
 * -jouer (class Partie)
 */

void vers_la_droite();
/*
 * Deplace tout les pions de la grille vers la droite de cette derniere
 comme si la gravité s'effectuée de gauche
 * à droite.
 * Cette fonction est appelée par :
 * -pivoter (class grille)
 */

void vers_la_gauche();
/*
 * Deplace tout les pions de la grille vers la gauche de cette derniere
 comme si la gravité s'effectuée de droite
 * à gauche.
 * Cette fonction est appelée par :
 * -pivoter (class grille)
 */
```

```
        void tourner_horaire();
        /*
         * Apres le deplacement des jetons, la grille tourne dans le sens horaire en
         inversant les lignes et les colonnes
         * Cette fonction est appelée par :
         * -pivoter (class grille)
         */

        void tourner_antihoraire();
        /*
         * Apres le deplacement des jetons, la grille tourne dans le sens
         antihoraire en inversant les lignes et les colonnes
         * Cette fonction est appelée par :
         * -pivoter (class grille)
         */

        void echanger(char forme_ori,char dest, int couleur_ori, int couleur_dest);
        /*
         * Echange les positions des jetons de deux joueurs. C'est à dire touts les
         jetons du joueurA vont aller à la place
         * de ceux du joueurB.
         * Cette fonction est appelée par :
         * -jouer (class partie)
         */

        int nb_coup_possible();
        /*
         * Revoi le nombre de case vide de la vide pour voir combien de placement il
         reste avant une partie nulle
         * Cette fonction est appelée par :
         * -tour_du_joueur (class partie)
         */

        void verification(jeton le_jeton, int joueur);
        /*
         * Appele de quatre fonction de vérification (verification_horizontale,
         verification_verticale,
         * verification_diagonale1 etverification_diagonale2)
         * Cette fonction est appelée par :
         * -tour_du_joueur (class partie)
         */

        void verification_victoire_horizontale(jeton le_jeton, int joueur);
        /*
         * Compte les alignement horizontaux de la grille du joueur qui joue
         * Cette fonction est appelée par :
         * -verification (class grille)
         */

        void verification_victoire_verticale(jeton le_jeton, int joueur);
        /*
         * Compte les alignement verticaux de la grille du joueur qui joue
         * Cette fonction est appelée par :
         * -verification (class grille)
         */

        void verification_victoire_diagonale_1(jeton le_jeton, int joueur);
        /*
         * Compte les alignement en diagonale droite(de haut en bas, de gauche a
         droite) de la grille du joueur qui joue
         * Cette fonction est appelée par :
         * -verification (class grille)
         */

        void verification_victoire_diagonale_2(jeton le_jeton, int joueur);
        /*
         * Compte les alignement en diagonale gauche(de haut en bas, de droite à
         gauche) de la grille du joueur qui joue
         * Cette fonction est appelée par :
         * -verification (class grille)
         */
```

```
void tout_verif();

int fin();
/*
 * Renvoi le resultat de la partie afin de designé le vainqueur
 * Cette fonction est appelée par :
 * -tour_du_joueur (class Partie)
 */

void impose_fin();
/*
 * met resultat a 1 pour finir une partie sans gagnant
 * Cette fonction est appelée par :
 * -victoire (class Partie)
 */

void nombre_joueur(int nombre);
/*
 * Donne une valeur a nb_joueur pour pouvoir l'utiliser dans la la class
 grille
 * Cette fonction est appelée par :
 * -joueurs_et_IA (class Partie)
 */

bool colonne_remplie(int col);
/*
 * Permet de savoir siune colonne est remplie ou pas
 * Cette fonction est appelée par :
 * -jouer (class Partie)
 */

int renvoi_nb_joueur();
/*
 * Renvoi le nombre de joueur s'ffrontant sur cette partie
 * Cette fonction est appelée par :
 * -menu_principal (class Partie)
 */

int renvoie_colonne();
/*
 * Renvoi le nombre de colonne de la grille de jeu
 * Cette fonction est appelée par :
 * -jouer (class Partie)
 */

int renvoie_ligne();
/*
 * Renvoi le nombre de ligne de la grille de jeu
 * Cette fonction est appelée par :
 * -
 */

void recupere_jeton_joueur(int joueur, jeton jeton_joueur);
/*
 * Recupere le jeton du joueur choisit pour exploiter ses données
 * Cette fonction est appelée par :
 * -menu_principal (class Partie)
 */

void sauvergarde(int joueur);
/*
 * Sauvegarde les données de la partie en cours dans un fichier texte afin
 de reprendre la
 * partie plus tard. Il n'y a pas de limitation sur le nom de sauvegarde ni
 sur le nombre
 * maximal
 * Cette fonction est appelée par :
 * -jouer (class Partie)
 */
```

```
                int charger(std::string nom);
                /*
                 * Charge une partie sauvegardé afin de la reprendre. L'utilisateur a juste
                 a mettre le nom sans le .txt et
                 * le chemin pour y acceder
                 * Cette fonction est appelée par :
                 * -menu_principal (class Partie)
                 */

                jeton renvoie_jeton_joueur(int joueur);
                /*
                 * Renvoie le jeton du joueur designé afin que Partie puisse l'exploité
                 * Cette fonction est appelée par :
                 * -charger_jeton_joueur (class Partie)
                 */

                 int renvoie_nb_alignement(int joueur);
                 /*
                  * Renvoie le nombre d'alignementeffectué par le joueur demandé
                  * Cette fonction est appelée par :
                  * -min_max (classe Intart)
                  * -min (classe Intart)
                  * -max (classe Intart)
                  */

                 void recopie(grille G_tmp);
                 /*
                  * Recopie entierement une grille dans une autre
                  * Cette fonction est appelée par:
                  * -min_max (classe Intart)
                  * -min (classe Intart)
                  * -max (classe Intart)
                  */

                 bool grille_remplie();
                 /*
                  * Test si une grille est remplie ou non
                  * Cette fonction est appelée par:
                  * -min (classe Intart)
                  * -max (classe Intart)
                  */

                 int renvoie_nb_jeton();
                 /*
                  * Renvoie le nombre de jetons present dans la grille de jeu
                  * Cette fonction est appelée par:
                  * -min (classe Intart)
                  * -max (classe Intart)
                  */

                 int renvoi_nb_al_necessaire();
                 /*
                  * Renvoie le nombre d'alignements necessaire pour gagner
                  * Cette fonction est appelée par:
                  * -min (classe Intart)
                  * -max (classe Intart)
                  */

                 int renvoie_nb_jeton_aligne(int joueur);
                 /*
                  * Renvoie le nombre maximal de jeton qui ont été aligné
                  * Cette fonction est appelée par:
                  * -min_max (classe Intart)
                  * -min (classe Intart)
                  * -max (classe Intart)
                  */

                 void annule_placement(int colonne);
};
#endif
```