```
//
//  ia.cpp
//  Morpion
//
//  Created by Corentin Bringer on 23/11/2020.
//

#include <ctime>
#include <cstdlib>
#include "ia.hpp"

/*
 Indique le gagnant ou continue la partie
 Parcourt la grille pour savoir si un joueur à gagné ou non

 012
 345
 678

 036
 147
 258

 048
 246

 @param char tabMorpion[9]
 @return char
 */
char isGameWin(char tabMorpion[9])
{
    char result = 'c';

    //Check horizontal for X
    for(int i = 0; i < 7; i += 3) {
        if(tabMorpion[i] == 'X' && tabMorpion[i + 1] == 'X' && tabMorpion[i
         + 2] == 'X') {
            result = 'X';
        }
    }

    //Check horizontal for O
    for(int i = 0; i < 7; i += 3) {
        if(tabMorpion[i] == 'O' && tabMorpion[i + 1] == 'O' && tabMorpion[i
         + 2] == 'O') {
            result = 'O';
        }
    }

    //Check vertical for X
    for(int i = 0; i < 3; i++) {
        if(tabMorpion[i] == 'X' && tabMorpion[i + 3] == 'X' && tabMorpion[i
         + 6] == 'X') {
            result = 'X';
        }
```

```
    }

    //Check vertical for O
    for(int i = 0; i < 3; i++) {
        if(tabMorpion[i] == 'O' && tabMorpion[i + 3] == 'O' && tabMorpion[i
         + 6] == 'O') {
            result = 'O';
        }
    }

    //Check diagonal for O
    for(int i = 0; i < 3; i++) {
        if(i == 0) {
            if(tabMorpion[0] == 'X' && tabMorpion[4] == 'X' &&
             tabMorpion[8] == 'X') {
                result = 'X';
            }
        }

        if(i == 2) {
            if(tabMorpion[2] == 'X' && tabMorpion[4] == 'X' &&
             tabMorpion[6] == 'X') {
                result = 'X';
            }
        }
    }

    //Check diagonal for O
    for(int i = 0; i < 3; i++) {
        if(i == 0) {
            if(tabMorpion[0] == 'O' && tabMorpion[4] == 'O' &&
             tabMorpion[8] == 'O') {
                result = 'O';
            }
        }

        if(i == 2) {
            if(tabMorpion[2] == 'O' && tabMorpion[4] == 'O' &&
             tabMorpion[6] == 'O') {
                result = 'O';
            }
        }
    }

    return result;
}

/*
 Regarde si des cases sont encore disponible
 @param char tabMorpion[9]
 @return bool
 */
bool isGameOver(char tabMorpion[9])
{
    bool result = false;
```

```
    for(int i = 0; i < 9; i++) {
        if(tabMorpion[i] != 'X' || tabMorpion[i] != 'O') {
            result = true;
        }
    }

    return result;
}

/*
 Verifie l'etat de jeu
 @param char tabMorpion[9]
 @return char
 */
char gameStateCheck(char tabMorpion[9])
{
    char result = 'c';

    result = isGameWin(tabMorpion);

    if(isGameOver(tabMorpion) == true) {
        result = 'd';
    }

    return result;
}

/*
 Verifie si la centre est disponible sinon ecrit O
 @param char tabMorpion[9]
 @return void
 */
void isCenterFree(char tabMorpion[9])
{
    if(tabMorpion[4] != 'X' || tabMorpion[4] != 'O') {
        tabMorpion[4] = 'O';
    }
}

/*
 Retourne une valeur comrpise entre 0 et 8
 @return int randomNb
 */
int random()
{
    int randomNb = 0;

    srand(time(0));
    randomNb = rand() % 8;

    return randomNb;
}

/*
```

```
 Regarde si une case est libre, si oui, la marqué par O
 @param char tabMorpion[9]
 @return void
 */
void iaRandomPlay(char tabMorpion[9])
{
    bool play = false;

    do {
        int randomNb = random();

        if(tabMorpion[randomNb] != 'X' && tabMorpion[randomNb] != 'O') {
            tabMorpion[randomNb] = 'O';
            play = true;
        }
    } while(play != false);
}

/*
 Si deux O sont aligné, compléter la case manquante si disponible
 @param char tabMorpion[9]
 @return void
 */
void iaPlayToWin(char tabMorpion[9])
{

}

/*
 Si deux X sont aligné, compléter la case manquante si disponible
 @param char tabMorpion[9]
 @return void
 */
void iaPlayToDefend(char tabMorpion[9])
{

}
```