Algo/JavaScript – USAL34

# Mini-projet : le jeu du morpion

**Objectifs du mini-projet**

Programmer le jeu du morpion à deux joueurs, où les joueurs tapent leurs coups alternativement.sur un clavier unique.

Le but du jeu du Morpion est d'aligner avant son adversaire 3 symboles identiques horizontalement, verticalement ou en diagonale. Chaque joueur a donc son propre symbole, généralement une croix pour l'un et un rond pour l'autre. Une partie typique devrait ressembler à l'écran à ceci :

```
---
---
---

Joueur1? 5
---
_x_
---

Joueur 2? 1
o__
_x_
---

Joueur 1? 7
o__
_x_
x__

Joueur 2? 3
o_o
_x_
x__

...
```

Plus précisément :

1. Avant chaque coup le programme affiche la grille : 3 lignes de 3 caractères : `'_'` pour la case vide, `'x'` pour un case cochée par le joueur 1, et `'o'` pour une case cochée par le joueur 2.
2. Ensuite le programme affiche le nom du joueur qui doit jouer (le joueur 1 commence) et lit le coup au clavier : un entier entre 1 et 9.
3. Si le coup est invalide pour une raison quelconque le coup n'est pas exécuté et c'est au encore au même joueur de jouer (retour à l'étape 1.).
4. Si le coup est valide et que la partie n'est pas finie le coup est exécuté (une case est cochée) et c'est au joueur suivant de jouer (retour à l'étape 1.).
5. Si le coup est autorisé et gagnant le programme s'arrêter (`exit(0);`) avec un message indiquant quel joueur a gagné.

**La grille du morpion en mémoire**
Le squelette de programme fourni est basé sur l'idée suivante : l'état du morpion se résume à l'état des 9 cases de la grille. À tout moment chaque case est dans l'un des 3 états suivants : vide, cochée `x` ou cochée `o`.

Pour représenter en mémoire la grille et les trois états possibles de chaque case on utilisera un tableau de caractères de 9 cases `grille[0]`...`grille[8]` de type `char` déclaré dans la fonction `main`.

```
int main() {
  char grille [] = {'_','_','_','_','_','_','_','_','_'};
```

Ce tableau représente les cases de la grille du morpion : `grille[0]` représente la case 1, `grille[1]` la case 2, et plus généralement la case `i` est représentée par `grille[i-1]`.

*Votre programme doit faire en sorte* qu'à tout moment le caractère contenu dans chaque variable correspond à l'état de la case correspondante (`'_'` si la case est vide, `'x'` si elle a été cochée par le joueur 1 et `'o'` si par le joueur 2). De cette façon pour afficher la grille il suffit d'afficher le contenu des variables, pour savoir l'état de la case i il suffit de regarder le contenu de la variable `grille[i-1]`.

**Traduction position dans la grill (x,y) ⇒ position dans le tableau (i)**
Considérons la grille 4x4 (`larg=4`) suivante, numérotée pour l'utilisateur (à partir de 1) et la case (4,3) ($4^e$ colonne, $3^e$ ligne) :

2 lignes complètes

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | '_' | 'x' | '_' | 'o' |
| 2 | 'x' | 'o' | '_' | '_' |
| 3 | '_' | 'x' | '_' | 'o' |
| 4 | 'x' | 'o' | '_' | '_' |

Cette grille est représentée en mémoire par un tableau à 16 cases :

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| _ | 'x' | '_' | 'o' | 'x' | 'o' | '_' | '_' | '_' | 'x' | '_' | 'o' | 'x' | 'o' | '_' | '_' |

la $4^e$ case de la $3^e$ ligne se situe dans le tableau à la case $(2 \times \texttt{larg} + 3)$. Plus généralement la case $(n, m)$ (colonne $n$ , ligne $m$) se situe à la case $(n-1) \times \texttt{larg} + m - 1$.

Petite précision : vous pouvez écrire des fonctions qui prennent le tableau en paramètre (à la place de `c1`...`c9`). *GROS AVANTAGE : vos fonctions peuvent modifier le contenu des cases du tableau passé en paramètre* (en réalité c'est l'adresse du tableau qui est envoyé à la fonction). On peut donc imaginer une nouvelle fonction :

```
void setCase(int i, char val, int[] g) {
  grille[i] = val;
}
```

telle que `setCase(casecochee-1,'x',grille)` met le caractère `'x'` dans la case numéro `casecochee-1` du tableau `grille`. Attention : les numéros de case commencent à 0.

Le squelette fourni contient plusieurs fonctions et procédures que vous devez compléter puis utiliser dans le `main`. N'hésitez pas à ajouter des fonctions supplémentaires si vous en avez le besoin.

- `BOOL ligneIdentique(char grille[], int ligne)` qui renvoie `TRUE` si la `ligne` du tableau `grille` est identique.
- `BOOL colonneIdentique(char grille[], int col)` qui renvoie `TRUE` si la `col` du tableau `grille` est identique.
- `BOOL diag1Identique(char grille[])` qui renvoie `TRUE` si la diagonale majeure du tableau `grille` est identique.
- `BOOL diag2Identique(char grille[])` qui renvoie `TRUE` si la diagonale mineure du tableau `grille` est identique.
- `BOOL testGagne(char grille[])` qui renvoie `TRUE` si le tableau `grille` est une position gagnante.

— `void afficheGrille(char grille[])` qui affiche les neufs caracteres du tableau `grille` comme une grille dans le terminal : 3 lignes de 3 caracteres.

Le déroulement du jeu est à programmer dans la boucle `while` de la fonction `main`, dont les étapes sont les suivantes :

1) Afficher la grille
2) Lire un entier au clavier
3) Redemander la case jusqu'a obtenir une case existantes et vide.
4) Maintenant on peut jouer
5) Si coup gagnant mettre a jour pasGagne (testGagne peut etre utile ici)
6) Si toutes les cases ont été jouées mettre a jour caseLibres
7) Changer de joueur

**Bonus**

1. Définissez une fonction `lireCoupMorpion` de lecture d'un entier au clavier qui redemande un coup tant que celui-ci est invalide. Quels arguments doit prendre cette fonction ? Que doit-elle retourner ? Pour redemander la lecture vous pouvez soit utiliser une boucle `while` (voir ci-dessous) soit simplement rappeler la fonction `lireCoupMorpion`.
2. Il s'agit maintenant de faire évoluer votre code (dans le `main` et dans `afficheGrille`) pour supporter plusieurs tailles de grille. La taille de la grille sera fixée avant de compiler en modifiant la valeur d'une variable (`largeurGrille` voir ci-dessous) au début du main.
   (a) Changez la déclaration du tableau `t` de la manière suivante (il se peut que l'option `-std=c99` soit nécessaire) :

   ```
   const int largeurGrille = 4;
   char t[largeurGrille*largeurGrille];
   ```

   L'idée est que pour changer de taille de grille il vous suffit de modifier le « `4` » ci-dessous par une autre taille pour que tout marche correctement. Il n'est pas demandé de programmer le test de victoire (mais vous pouvez essayé si tout le reste est fini).
   (b) Reprogrammez l'initialisation de la grille (toutes les cases mises à `'_'`) pour qu'elle s'adapte à la largeur `largeurGrille`.
   (c) Ajouter un argument `int larg` aux procédures et fonctions de votre code. Cet argument représente le nombre de lignes (et de colonnes) de la grille. Adaptez votre code à ce nouveau paramètre.

## ▶ Correction

```
/*@debmorpion@*/
#include<stdlib.h>
#include"inout.h"
#define BOOL int
#define TRUE 1
#define FALSE 0

BOOL ligneIdentique(char grille[], int ligne)
{

  if (grille[3*(ligne-1)] ==grille[3*(ligne-1)+1] &&  grille[3*(ligne-1)] ==grille[3*(ligne
    return TRUE;
  return FALSE;
}

BOOL colonneIdentique(char grille[], int col)
```

```
{
  if (grille[col-1] ==grille[3+col-1] &&  grille[col-1] ==grille[6+col-1] && grille[col-1]
    return TRUE;
  return FALSE;
}

BOOL diag1Identique(char grille[])
{
  if (grille[0] ==grille[4] &&  grille[0] ==grille[8] && grille[0]!='_')
    return TRUE;
  return FALSE;
}

BOOL diag2Identique(char grille[])
{
  if (grille[2] ==grille[4] &&  grille[2] ==grille[6]&& grille[2]!='_')
    return TRUE;
  return FALSE;
}

/* Teste si la grille decrite par les 9 caracteres est une position gagnante. */
BOOL testGagne(char grille[]){
/* SOLUTION:  return TRUE; // INTENTIONNELLEMENT FAUX, a vous de l'ecrire*/
  int i;
    if (diag1Identique(grille)==1)
      return TRUE;
    if (diag2Identique(grille)==1)
      return TRUE;
    for (i =1; i<=3;i++)
      {
        if (ligneIdentique(grille,i)==1)
          return TRUE;
        if (colonneIdentique(grille,i)==1)
          return TRUE;
      }
    return FALSE;
}


/* Affiche les neufs caracteres comme une grille dans le terminal: 3 lignes de 3 caracteres
void afficheGrille(char grille[]){
  int i,j;
  for (i=0;i<3;i++)
    {
      for (j=0;j<3;j++)
        ecrireChar(grille[i*3+j]);
      ecrireSautDeLigne();
    }
}


int main() {
  // represente la grille du jeu
  char grille[9] = {'_','_','_','_','_','_','_','_','_'};
  // contiendra le numÃ©ro dÃ©signÃ© par le joueur qui joue
  int casecochee;
  // doit devenir faux lorsque une victoire est detectee, afin d'arreter la boucle while c
  BOOL pasGagne=TRUE;
  //doit devenir faux quand les cases ont toutes Ã©tÃ© cochÃ©es, afin d'arreter la boucle
  BOOL caseLibres=TRUE;
  // L'entier correspond au joueur dont c'est le tour
```

```
  char joueur ='x';
  // Boucle principale: on en sort lorsqu'un jouer gagne.
  int nbCoupsjoues =0;
  while(pasGagne && caseLibres) {

    // 1) afficher la grille
    afficheGrille(grille);
    // 2) lire un entier au clavier
    ecrireString("Joueur ");
    ecrireChar(joueur);
    ecrireString(" ? ");
    casecochee = lireInt();

    // 3) Redemander la case jusqu'a obtenir une case existantes et vide.
    if (casecochee <1 || casecochee >9 || grille[casecochee -1] != '_')
      {
        if (casecochee<0 || casecochee>9) {
          ecrireString("\nCeci n'est pas un numÃ©ro de case.");
        }
        else {
          ecrireString("\nCase interdite.");
        }
        ecrireString("recommencez ");
        casecochee = lireInt();
      }

    // Maintenant on peut jouer
    grille[casecochee-1]=joueur;
    nbCoupsjoues++;

    // 4) si coup gagnant mettre a jour pasGagne
    // (testGagne peut etre utile ici)
    if (testGagne(grille) ==1)
      pasGagne = FALSE;

    // 5) Si toutes les cases ont Ã©tÃ© jouÃ©es mettre a jour caseLibres
    if (nbCoupsjoues==9)
      caseLibres=FALSE;
    // 6) changer de joueur (joueur devient 'x' (si 'o') et 'o' (si 'x'))
    if (caseLibres ==1 && pasGagne==1)
      {
        if(joueur=='x')
          joueur = 'o';
        else
          joueur = 'x';
      }
/* FIN SOLUTION */
  }
  // Afficher une derniere fois la grille et sortir
  if (pasGagne == 0)
    {
      ecrireString("victoire du joueur ");
      ecrireChar(joueur);
      ecrireString(" !\n");
    }
  else
    {
      ecrireString("Partie terminÃ©e, aucun vainqueur \n");
    }

  afficheGrille(grille);
```

```
}
```