Programmation Impérative
TP BILAN

1) Résolution de "faire un jeu de Morpion"

a) Énoncé

Le jeu du Morpion est un jeu qui se joue à deux joueurs sur un damier carré (n*n). Chaque joueur joue en alternance en inscrivant son symbole (un rond ou une croix suivant le joueur) dans une case libre (sans rond, ni croix). Initialement, toutes les cases du damier sont libres.

Le jeu s'arrête dès que l'un des joueurs aligne n symboles identiques horizontalement, verticalement ou en diagonale, ou lorsque toutes les cases sont occupées.
On dispose des déclarations suivantes :

```
KMAX : constante entier est 10          (* taille max du damier *)
Type SYMBOLE est (LIBRE,ROND,CROIX)     (* contenu d'une case du damier *)
Type DAMIER est tableau (1..KMAX, 1..KMAX) de SYMBOLE  (* un damier *)
Type JOUEUR est (JROND, JCROIX)         (* deux joueurs seulement *)
Type ETAT_JEU est (EN_COURS, GAGNE, NUL)

le_damier : DAMIER                      (* espace de jeu *)
le_n : INTEGER                          (* dimension r éelle du damier.
                                        Entre 3 et KMAX *)
le_joueur : JOUEUR                      (* le joueur courant *)
l_etat : ETAT_JEU                       (* l'etat courant du jeu *)
```

b) R0 + Tests

Effectuer un programme qui permet à deux joueurs de jouer au Morpion sur un damier dont ils définiront la taille.
Tests : Partie de Morpion 3*3 gagnée par le joueur Croix
Partie de Morpion 6*6 gagnée par le joueur Rond
Partie de Morpion 4*4 match nul

c) R1

Lire le nombre de cases souhaitées pour le damier
Initialiser le jeu
Jouer
Afficher la fin du jeu

d) R2

(* Lire le nombre de cases souhaitées pour le damier *)

```
FAIRE
      ECRIRE("Veuillez saisir le nombre de case que vous souhaitez pour votre
               damier (3<nbCases<",KMAX,")");
      LIRE(le_n);
TANT QUE le_n <3 ET le_n > KMAX;
```

(* Initialiser le jeu *)

```
(* Préconditions : le_n >= 3 ET le_n<= KMAX
 * Post conditions : POUR TOUT i DANS 1..le_n
 *                          POUR TOUT j DANS 1..le_n
 *                                  le_damier(i)(j) = LIBRE
*)
initialiser_jeu(le_damier, le_n, JCROIX);
```

(* Jouer *)

```
(* Préconditions : l_etat = EN_COURS *)
jouer(le_damier, le_n, l_etat, le_joueur);
```

(* Afficher la fin du jeu *)
(Préconditions : l_etat = GAGNE OU l_etat = NUL *)*
```
afficher_fin_de_jeu(le_damier, le_n, le_joueur, l_etat);
```

2) Résolution de "initialiser le jeu"

a Énoncé

Initialiser l'ensemble du damier (n*n), l'ensemble de ses cases, à l'état LIBRE.

e) R0 + Tests

On choisit d'écrire ce sous problème sous forme de sous programme.
```
-- procedure : initialiser_jeu
-- Permet d'initialiser le damier en mettant dans toutes ces cases le symbole "LIBRE"
-- Paramètres le_damier : Mode Données/Résultats
--                        Type DAMIER
--                        L'espace de jeu à initialiser
--                le_n  :  Mode Données
--                         Type INTEGER
--                         La taille réelle du damier
--            le_joueur :  Mode Données
--                         Type JOUEUR
--                         Le joueur courant
-- Préconditions : le_n >= 3 ET le_n<= KMAX
-- Post conditions : POUR TOUT i DANS 1..le_n
--                       POUR TOUT j DANS 1..le_n
--                             le_damier(i)(j) = LIBRE

procedure initialiser_jeu(le_damier: IN OUT DAMIER; le_n: INTEGER; le_joueur: JOUEUR)
```

Tests : initialisation d'un damier 3*3
initialisation d'un damier 4*4
initialisation d'un damier 10*10

f) R1

(* initialiser le jeu *)
```
POUR i ALLANT DE 1 À le_n FAIRE
      POUR j ALLANT DE 1 À le_n FAIRE
            le_damier(i)(j) <- LIBRE
      FPOUR
FPOUR
```

3) Résolution de "afficher jeu"

a Énoncé

Afficher l'état courant du jeu sous forme de damier comportant un "." pour les cases libres, une "X" pour les cases occupées par le joueur CROIX et un "O" pour les cases occupées par le joueur ROND.

g) R0 + Tests

On choisit d'écrire ce sous problème sous forme de sous programme.

```
--------------------------------------------------------------------------------
-- procedure : afficher_jeu
-- Permet d'afficher l'état courant du jeu à l'aide de caractères ASCII
-- Exemple d'affichage pour un morpion 5x5
--
--              0   1   2   3   4
--          0   .   .   .   .   .
--          1   .   .   X   X   .
--          2   .   .   .   X   .
--          3   .   .   .   .   .
--          4   O   O   O   X   O
-- Paramètres le_damier : Mode Données
--                        Type DAMIER
--                        L'espace de jeu à initialiser
--                le_n  : Mode Données
--                        Type INTEGER
--                        La taille réelle du damier
--            le_joueur : Mode Données
--                        Type JOUEUR
--                        Le joueur courant
-- Préconditions : le_n >= 3 ET le_n<= KMAX
-- Postcondition : ____
--------------------------------------------------------------------------------

procedure afficher_jeu(le_damier: DAMIER; le_n: INTEGER; le_joueur: JOUEUR)
```

h) R1

```
(* AFFICHAGE DE LA PREMIÈRE LIGNE CONTENANT LES NUMÉROS DE COLONNES *)
POUR i ALLANT DE 1 À le_n FAIRE
      ECIRE(i-1,"   ");
FPOUR
(* AFFICHAGE DU RESTE DU DAMIER *)
POUR i ALLANT DE 1 À le_n FAIRE
      (* NUMÉRO DE LIGNE *)
      ECRIRE(i,"   ")
      POUR j ALLANT DE 1 À le_n FAIRE
            SI le_damier(i)(j) = ROND ALORS
                  ECRIRE("O")
            SINON SI le_damier(i)(j) = CROIX ALORS
                  ECRIRE("X")
            SINON
                  ECRIRE(".")
            FSI
      FPOUR
FPOUR
```

4) Résolution de "jouer"

a Énoncé

Permettre à un joueur d'effectuer son tour, tester si son tour change quelque chose dans l'état de la partie (GAGNE, NUL ou toujours EN_COURS).

i) R0 + Tests

On choisit d'écrire ce sous problème sous forme de sous programme.

```
-------------------------------------------------------------------------------
-- procedure : jouer
-- Permet d'effectuer un tour de jeu à savoir demander au joueur courant
      "le_joueur" la case dans laquelle il veut jouer puis de placer le symbole
      correspond au joueur en cours dans la case correspondante.
-- Enfin la procédure teste si la nouvelle situation modifie l'état de la partie
      si c'est le cas elle change l'état sinon elle change le joueur courant
-- Paramètres le_damier : Mode Données/Résultats
--                        Type DAMIER
--                        L'espace de jeu à initialiser
--              l_etat  : Mode Données/Résultats
--                        Type ETAT_JEU
--                        L'état courant du jeu
--            le_joueur : Mode Données/résultat
--                        Type JOUEUR
--                        Le joueur courant
-- Préconditions : l_etat = EN_COURS
-- Postcondition : ____
-------------------------------------------------------------------------------

procedure jouer(le_damier: IN OUT DAMIER; le_n: Integer; l_etat: IN OUT ETAT_JEU;
le_joueur: IN OUT JOUEUR)
```

j) R1

1. Attribution du symbole à insérer selon le joueur courant
2. Saisie conviviale et fiable du numéro de ligne et colonne où l'utilisateur veut jouer
3. Affectation du symbole à la case souhaitée
4. Test de l'état du jeu après le tour
5. Modification de l'état du jeu ou du joueur suivant

Avec les déclarations suivantes :

```
ligne: Integer;          -- NUMÉRO DE LIGNE SAISI PAR L'UTILISATEUR
colonne: Integer;        -- NUMÉRO DE COLONNE SAISI PAR L'UTILISATEUR
symbole: SYMBOLE;        -- SYMBOLE À INSÉRER DANS LE DAMIER
next_etat: ETAT_JEU;     -- ETAT DU JEU APRÈS LE TOUR
```

Analyse informelle :

1. **Structure de contrôle :** SI
   Test si le_joueur = JROND alors symbole := ROND sinon symbole := CROIX
2. **Structure de contrôle :** REPETER
   Répéter la saisie tant que *ligne* < *1* **et** *ligne* > *le_n* **et** *colonne* < *1* **et** *colonne* > *le_n* **et** *le_damier(ligne)(colonne) /= LIBRE*
3. Simple affectation
4. On doit tester tous les cas possibles d'alignements de symbole donc on a choisi de traiter ce sous problème dans un sous programme.
5. **Structure de contrôle :** SI
   Test si le nouvel état est différent de EN_COURS, si c'est le cas alors on affecte le nouvel état à l'état du jeu sinon on modifie le joueur courant en lui affectant l'autre joueur.

```
-- ATTRIBUTION DU SYMBOLE À INSÉRER SELON LE JOUEUR COURANT
SI le_joueur = JROND ALORS
      symbole := ROND;
```

```
SINON
      symbole := CROIX;
FIN SI

-- SAISIE DE LA POSITION DU PROCHAIN COUP DU JOUEUR COURANT DE MANIÈRE CONVIVIALE ET
FIABLE
FAIRE
      ECRIRE("JOUEUR ",symbole);
      ECRIRE ("Veuillez saisir le numéro de ligne où vous souhaitez jouer : ");
      LIRE(ligne);
      ECRIRE ("Veuillez saisir le numéro de colonne où vous souhaitez jouer : ");
      LIRE(colonne);
TANT QUE ligne < 1 ET colonne < 1 ET ligne > le_n ET colonne > le_n ET
le_damier(ligne)(colonne) = LIBRE;

-- AFFECTATION DU SYMBOLE DU JOUEUR COURANT À LA CASE DEMANDÉE
le_damier(ligne)(colonne) := symbole;

-- TEST DE L'ÉTAT DU JEU APRÈS LE TOUR -> SOUS PROGRAMME
next_etat := how_etat_is(le_damier,le_n,symbole);

-- MODIFICATION DE L'ÉTAT DU JEU OU DU JOUEUR SUIVANT
SI next_etat /= EN_COURS ALORS
      l_etat := next_etat;
SINON
      SI le_joueur = JROND ALORS
            le_joueur := JCROIX;
      SINON
            le_joueur := JROND;
      FIN SI
FIN SI
```

5) Résolution de "test de l'état courant"

a Énoncé

On doit tester tout le damier pour savoir si il y a une situation gagnante, une situation nulle ou une partie en cours.
Une situation gagnante se définit comme un alignement (vertical/horizontal/diagonal) de **le_n** symbole pour un damier **le_n*le_n**.
Une situation nulle se définit par un damier rempli (aucune case LIBRE) sans présenter de situation gagnante.
Une situation en cours se définit par un damier partiellement rempli dans lequel il reste des cases non affectées.

k) R0 + Tests

On choisit d'écrire ce sous problème sous forme de sous programme.

```
--------------------------------------------------------------------------------
-- fonction : how_etat_is
-- Permet de tester si la situation courante du jeu/du damier est une situation
      gagnante, nulle ou en cours à savoir qu'une situation gagnante est une
      sitation où "le_n" symboles identiques sont alignés horizontalement
      verticalement ou  diagonalement et une situation est une situation où
      toutes les cases du     damiers sont utilisées sans présenter une situation
      gagnante.
```

```
-- Paramètres le_damier : Mode Données
--                        Type DAMIER
--                        L'espace de jeu à initialiser
--                le_n  : Mode Données
--                        Type INTEGER
--                        La taille réelle du damier
--           le_symbole : Mode Données
--                        Type SYMBOLE
--                        Le symbole joueur courant
-- Résultat : Type ETAT_JEU
--            GAGNE si le damier comporte un alignement de "le_n" symboles
                identiques
--            NUL si toutes les cases sont utilisées mais aucune situation    --
                gagnante
--            EN_COURS si ni NUL ni GAGNE
-- Préconditions : le_n >= 3 ET le_n<= KMAX
-- Postcondition : ____
--------------------------------------------------------------------------------

function how_etat_is(le_damier: DAMIER; le_n: Integer; le_symbole: SYMBOLE)
return ETAT _JEU
```

Tests : Damier vide -> EN_COURS
Alignement horizontal -> GAGNE
Alignement vertical -> GAGNE
Alignement diagonal -> GAGNE
Damier plein sans alignement -> NUL
Damier partiellement rempli -> EN COURS

l) R1

1. Test de l'alignement des symboles sur les lignes et les diagonales
2. Test de l'alignement des symboles sur les colonnes
3. Test de l'état nul (rempli sans cas gagnant)

Avec les déclarations suivantes :

```
next_etat: ETAT_JEU;
cpt_symboles_alignes_lignes: Integer;
cpt_symboles_alignes_colonnes: Integer;
cpt_symboles_alignes_diagonales1: Integer;
cpt_symboles_alignes_diagonales2: Integer;
cpt_symboles_total: Integer;
I: Integer;
J: Integer;
```

Analyse informelle :

1. **Structure de contrôle :** TANT QUE

Tant que le parcours n'est pas fini et que l'état est toujours EN_COURS on parcourt toutes les colonnes de chaque ligne. Pendant ce parcours on ne comptabilise que le symbole qui vient d'être inséré dans le damier car il n'y a que lui qui peut avoir fait changer l'état courant du jeu.

Pour chaque case on vérifie si elle

- n'est pas LIBRE -> incrémentation du compteur *cpt_symboles_total*
- appartient a une des deux diagonales et comporte le symbole recherché
  -> incrémentation *cpt_symboles_alignes_diagonales2* ou *diagonales1*

- comporte le symbole recherché -> incrémentation *cpt_symboles_alignes_lignes*

Avant chaque changement de ligne on teste si il y a un alignement et si c'est le cas on change l'état *next_etat* sinon on remet à zéro le compteur ligne et on change de ligne.

On teste pour finir si il y a alignement sur une des deux diagonales auquel cas on modifie l'état du jeu.

2. **Structure de contrôle :** TANT QUE
Tant que le parcours n'est pas fini et que l'état est toujours EN_COURS on parcourt toutes les lignes de chaque colonne.
Pour chaque case on vérifie si elle comporte le symbole recherché -> incrémentation *cpt_symboles_alignes_colonnes*.
Avant chaque changement de colonne on teste si il y a eu un alignement si c'est le cas on change l'état sinon on remet à zéro le compteur colonne et on change de colonne.

3. **Structure de contrôle :** SI
Test si le compteur *cpt_symboles_total* est égal au nombre de cases totales (**le_n*le_n**)
Si c'est le cas -> état devient NUL
SINON etat reste EN_COURS

m)R2

1. Test de l'alignement des symboles sur les lignes et les diagonales
Test si case courante LIBRE pour etat NUL
Test si case courante appartient aux diagonales et comporte symbole recherché
Test si case comporte symbole recherché pour les lignes

2. Test de l'alignement des symboles sur les colonnes
Test si case courante comporte symbole recherché pour les colonnes

3. Test de l'état nul (rempli sans cas gagnant)
Test si compteur est égal au nombre de cases totales

```
fonction how_etat_is(le_damier: DAMIER; le_n: entier; le_symbole: SYMBOLE)
retourn ETAT _JEU est

-- TEST DE L'ALIGNEMENT DES SYMBOLES SUR LES LIGNES ET LES DIAGONALES
TANT QUE next_etat = EN_COURS and I <= le_n FAIRE
      TANT QUE next_etat = EN_COURS and J <= le_n FAIRE
            -- TEST SI CASE LIBRE POUR TEST D'ÉTAT NUL
                  SI le_damier(I)(J) /= LIBRE ALORS
                        cpt_symboles_total := cpt_symboles_total +1;
                  SINON
                        rien
                  FSI

            -- TESTS POUR LES DIAGONALES
                  SI I = J and le_damier(I)(J) = le_symbole ALORS
                        -- DIAGONALE DE HAUT GAUCHE À BAS DROIT
                        cpt_symboles_alignes_diagonales1 :=
                              cpt_symboles_alignes_diagonales1 +1;
```

```
                  SINON SI J = (le_n-I)  ALORS
                        -- DIAGONALE DE HAUT DROIT À BAS GAUCHE
                        cpt_symboles_alignes_diagonales2 :=
                              cpt_symboles_alignes_diagonales2 +1;
                  FSI

            -- TEST POUR LES LIGNES
                  SI le_damier(I)(J) = le_symbole ALORS
                        cpt_symboles_alignes_lignes :=
                              cpt_symboles_alignes_lignes +1;
                  SINON
                        rien
                  FSI
            J := J+1; -- INCRÉMENTATION NUMÉRO DE COLONNE
      FTANT QUE
      -- FIN DU PARCOURS DE LA LIGNE COURANTE DU DAMIER OU SITUATION GAGNANTE TROUVÉE

      -- TEST DU NOMBRE DE SYMBOLE ALIGNÉS SUR LA LIGNE PARCOURUE
      SI cpt_symboles_alignes_lignes = le_n ALORS
            next_etat := GAGNE;
      SINON
            cpt_symboles_alignes_lignes := 0;
      FSI
      J := 1; -- REMISE À ZÉRO DU NUMÉRO DES COLONNES POUR PASSER À LA LIGNE SUIVANTE
      I := I+1;   -- INCRÉMENTATION DU NUMÉRO DE LIGNE
FTANT QUE

-- TEST DE SITUATION GAGNANTE SUR LES DIAGONALES SI SITUATION PAS DÉJÀ GAGNANTE
      SI next_etat = EN_COURS and (cpt_symboles_alignes_diagonales1 = le_n or
cpt_symboles_alignes_diagonales2 = le_n) ALORS
            next_etat := GAGNE;
      SINON
            rien
      FSI

-- TESTS DES COLONNES
      TANT QUE next_etat = EN_COURS and J <= le_n FAIRE
            TANT QUE next_etat = EN_COURS and I <= le_n FAIRE
                  -- TESTS POUR LES COLONNES
                        SI le_damier(I)(J) = le_symbole ALORS
                              cpt_symboles_alignes_lignes :=
                                    cpt_symboles_alignes_colonnes +1;
                        SINON
                              rien
                        FSI

                  I := I+1;
            FTANT QUE

            -- TEST DU NOMBRE DE SYMBOLE ALIGNÉS SUR LA COLONNE PARCOURUE
            SI cpt_symboles_alignes_colonnes = le_n ALORS
                  next_etat := GAGNE;
            SINON
                  cpt_symboles_alignes_colonnes := 0;
            FSI

            I := 1; -- REMISE À ZÉRO DU NUMÉRO DES DE LIGNE POUR PASSER À LA COLONNE
SUIVANTE
            J := J+1;   -- INCRÉMENTATION DU NUMÉRO DE COLONNE
      FTANT QUE
```

```
-- TEST POUR L'ÉTAT NUL
      SI next_etat = EN_COURS and cpt_symboles_total = (le_n*le_n) ALORS
            next_etat := NUL;
      SINON
            rien
      FSI

retourne next_etat;
```

6) Résolution de "afficher fin de jeu"

a Énoncé

On doit afficher le damier comportant soit un alignement soit une partien nulle et afficher le joueur gagnant si il y en a un ou un message témoignant du match nul.

n) R0 + Tests

On choisit d'écrire ce sous problème sous forme de sous programme.

```
--------------------------------------------------------------------------------
-- procedure : afficher_fin_de_jeu
-- Permet d'afficher l'état du jeu lorsqu'il est fini
-- Paramètres le_damier : Mode Données
--                        Type DAMIER
--                        L'espace de jeu à initialiser
--                le_n  : Mode Données
--                        Type INTEGER
--                        La taille réelle du damier
--            le_joueur : Mode Données
--                        Type JOUEUR
--                        Le joueur courant
--              l_etat  : Mode Données/Résultats
--                        Type ETAT_JEU
--                        L'état courant du jeu
-- Préconditions : le_n >= 3 ET le_n<= KMAX ET l_etat /= EN_COURS
-- Postcondition : ____
--------------------------------------------------------------------------------

procedure afficher_fin_de_jeu(le_damier: DAMIER; le_n: INTEGER; le_joueur:
JOUEUR, l_etat: ETAT_JEU)
```

o) R1

1. Affichage damier classique
2. Affichage du message de fin de jeu

Analyse informelle :

1. Appel de afficher_jeu(le_damier,le_n,le_joueur)
2. **Structure de contrôle** : SI
   Test si *l_etat* est égal à NUL afficher "MATCH NUL ! "
   Sinon afficher "Le joueur " + le_joueur + " a gagné la partie !"

```
-- AFFICHAGE DU DAMIER CLASSIQUE
      afficher_jeu(le_damier,le_n,le_joueur);
```

```
-- AFFICHAGE DU MESSAGE
      SI l_etat = NUL ALORS
                  ECRIRE("MATCH NUL ! ");
      SINON
            ECRIRE ("Le joueur ",le_joueur, " a gagné la partie !");
      FSI
```