**Programmation Impérative**

**TP BILAN**

1) Résolution de "faire un jeu de Morpion"

**a)** **Énoncé**

**b)** **R0 + Tests**

**c)** **R1**

**d)** **R2**

2) Résolution de "initialiser le jeu"

**a)** **Énoncé**

**b)** **R0 + Tests**

**c)** **R1**

3) Résolution de "afficher jeu"

**a)** **Énoncé**

**b)** **R0 + Tests**

**c)** **R1**

4) Résolution de "jouer"

**a)** **Énoncé**

**b)** **R0 + Tests**

**c)** **R1**

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

**a)** **Énoncé**

**b)** **R0 + Tests**

**c)** **R1**

**d)** **R2**

6) Résolution de "afficher fin de jeu"

**a)** **Énoncé**

**b)** **R0 + Tests**

**c)** **R1**

1. Résolution de "faire un jeu de Morpion"
2. **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 ́eelle du damier. Entre 3 et KMAX \*)

le\_joueur : JOUEUR (\* le joueur courant \*)

l\_etat : ETAT\_JEU (\* l’etat courant du jeu \*)

1. **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

1. **R1**

Lire le nombre de cases souhaitées pour le damier

Initialiser le jeu

Jouer

Afficher la fin du jeu

1. **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);

1. Résolution de "initialiser le jeu"
2. **Énoncé**

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

1. **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

1. **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

1. Résolution de "afficher jeu"
2. **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.

1. **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)

1. **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

1. Résolution de "jouer"
2. **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).

1. **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)

1. **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

1. Résolution de "test de l'état courant"
2. **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.

1. **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

1. **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

1. **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;

1. Résolution de "afficher fin de jeu"
2. **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.

1. **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)

1. **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