TP BILAN : MORPION

PLAN :

1)Résolution du Morpion (pb)

a) Enoncé

Permettre à deux joueur de jouer au morpion.

b) R0

Sémantique : Les deux joueurs jouent tour à tour en remplissant une case avec une croix ou un rond et on affiche le résultat de la partie.
Préconditions : aucunes
Postconditions : les 9 cases du damier sont remplies ou trois mêmes symboles sont alignés
tests : partie nulle
partie gagnée par un des joueurs

c) R1

```
Procedure Morpion est
Debut
      initialiser_jeu(le_damier, le_joueur)              (ss-pb 1)
      afficher_jeu(le_damier, le_joueur)                 (ss-pb 2)
      repeter
            jouer(le_damier, le_joueur, l_etat)          (ss-pb 3)
            afficher_jeu(le_damier, le_joueur)
      jusqu'a l_etat /= EN_COURS
      afficher l'etat du jeu en fin de partie            (ss-pb 4)
Fin Morpion
```

avec comme types et variables :

```
Type CASE est (Vide, Rond, Croix)
Type DAMIER est matrice 3x3 de CASE
Type JOUEUR est (Rond, Croix)
Type ETAT_JEU est (EN_COURS, GAGNE, NUL)
le_damier : DAMIER
le_joueur : JOUEUR
l_etat : ETAT_JEU;
```

2)Résolution de initialiser_jeu (ss-pb 1)

a) Enoncé

Initialiser le damier et le joueur.

b) R0

(* procédure initialiser_jeu
sémantique : initialiser le damier (matrice 3*3) avec des cases vides et permettre au premier joueur de choisir entre croix et rond.
données : aucune
données/résultats : aucun
résultats : Fle_damier le damier et Fle_joueur le joueur;
pré-conditions : aucune
post-conditions : aucune
*)

```
procedure initialiser_jeu ((*R*) Fle_damier : entier, (*R*) Fle_joueur: vecteur)
```

c) R1

```
(*initialiser_jeu*)
      initialiser damier      (ss-pb 1.1)
      initialiser joueur      (ss-pb 1.2)
```

d) R2

```
(*initialiser_jeu*)
      (*initialiser damier*)
      Pour i de 1 à 9 faire
            Fle_damier(i) <- Vide
      Fin pour
      (*initialiser joueur*)
      Ecrire("Que choisit le premier joueur? Rond/Croix")
      Lire(Fle_joueur)
```

3)Résolution de afficher_jeu (ss-pb 2)

a) Enoncé

Afficher le damier en code ASCII.

b) R0

(* procédure afficher_jeu
sémantique : afficher le damier grâce à des lettres et symboles.
données : Fle_damier le damier, Fle_joueur le joueur.
données/résultats : aucun
résultats : aucun
pré-conditions : aucune
post-conditions : aucune
*)

```
procedure afficher_jeu ((*D*) Fle_damier : entier, (*D*) Fle_joueur: vecteur)
```

c) R1

```
(*afficher jeu*)
```

```
      convertir damier en symboles       (ss-b 2.1)
      afficher ligne par ligne, symbole par symbole le contenu des cases
      (ss-pb 2.2)
```

d) R2

```
(*afficher jeu*)
      (*convertir damier en symboles*)
      Pour i de 1 à 9 faire
            Cas Fle_damier(i) de
                  Rond:damier_carac(i) <- 'o'
                  Croix:damier_carac(i) <- 'x'
                  Autres:damier_carac(i) <- ' '
      Fin pour
      (*afficher ligne par ligne*)
      Pour i de 0 à 2 faire
            afficher case par case       (ss-pb 2.2.1)
            nouvelle_ligne
            écrire("-- - --")
            nouvelle_ligne
      Fin pour
```

avec comme déclarations :

```
type matrice est tableau (1..9) de caractères
variable damier_carac : matrice
```

e) R3

```
(*afficher case par case*)
Pour j de 1 à 3 faire
      écrire(' ')
      écrire(damier_carac(i+j))
      écrire('|')
Fin pour
```

4)Résolution de jouer (ss-pb 3)

a) Enoncé

Un joueur joue sur une case, on affiche le damier modifié et on détermine si la partie est terminée

b) R0

(* procédure jouer
sémantique : demander au joueur sur quelle case il joue et déterminer si la partie est terminée.
données : Fle_joueur le joueur
données/résultats : Fle_damier le damier, Fl_etat l'état
résultats : aucun
pré-conditions : aucune
post-conditions : aucune
*)

```
procedure jouer ((*D/R*) Fle_damier : entier, (*D*) Fle_joueur: vecteur, (*D/R*)
Fl_etat : ETAT_JEU)
```

c) R1

```
(*jouer*)
      compter le nombre de coups joués                    (ss-pb 3.1)
      demander au joueur sur quelle case il veut jouer    (ss-pb 3.2)
      modifier la case correspondante                     (ss-pb 3.3)
      déterminer l'état de la partie                      (ss-pb 3.4)
      changer de joueur si la partie doit continuer       (ss-pb 3.5)
```

d) R2

```
(*jouer*)
      (*compter le nombre de coups joués*)
```

```
nombre_de_coups <- 1    (*on compte le coups qui va etre joué*)
Pour i de 1 à 9 faire
      Si /Fle_damier(i)=Vide alors nombre_de_coups <- nombre_de_coups+1
      Sinon rien
      fin si
fin pour
(*demander au joueur sur quelle case il veut jouer*)
écrire("Au tour des")
écrire(chaine(le_joueur))
écrire("Sur quelle case voulez-vous jouer? Entrez un nombre entre 1 et 9")
(*modifier la case correspondante*)
le_damier(lire(case_a_jouer))<- le_joueur
(*augmenter le nombre de coups joués*)
nombre_de_coups <- nombre_de_coups+1
(*déterminer l'état de la partie*)
éditer l'état de la partie si elle est gagnée        (ss-pb 3.4.1)
éditer l'état de la partie si elle est nulle         (ss-pb 3.4.2)
(*changer de joueur si la partie doit continuer*)
Si l_etat=EN_COURS alors Si le_joueur=Rond alors le_joueur <- Croix
                   Sinon le_joueur <- Rond
                   Fin si
Sinon rien
Fin si
```

avec comme déclaration :

```
variable nombre_de_coups : entier
```

e) R3

```
(*déterminer l'état de la partie*)
      (*éditer l'état de la partie si elle est gagnée*)
      Si ((le_damier(1)/=Vide et le_damier(1)=le_damier(2) et ledamier(2)=le_damier(3))
       ou(le_damier(4)/=Vide et le_damier(4)=le_damier(5) et le damier(5)=le_damier(6))
       ou(le_damier(7)/=Vide et le_damier(7)=le_damier(8) et le damier(8)=le_damier(9))
       ou(le_damier(1)/=Vide et le_damier(1)=le_damier(5) et le damier(5)=le_damier(9))
       ou(le_damier(7)/=Vide et le_damier(7)=le_damier(5) et le damier(5)=le_damier(3))
       ou(le_damier(1)/=Vide et le_damier(1)=le_damier(4) et le damier(4)=le_damier(7))
       ou(le_damier(2)/=Vide et le_damier(2)=le_damier(5) et le damier(5)=le_damier(8))
       ou(le_damier(3)/=Vide et le_damier(3)=le_damier(6) et le damier(6)=le_damier(9)))
      alors l_etat <- GAGNE
      (*éditer l'état de la partie si elle est nulle*)
      Sinon si nombre_de_coups=9 alors l_etat <- NUL
            sinon rien
            fin si
      fin si
```

5)Résolution de afficher l'etat du jeu en fin de partie (ss-pb 4)

a) Enonce

afficher l'état du jeu en fin de partie et en cas de victoire, afficher qui a gagné.

b) R0=Enonce

c) R1

```
Si l_etat=NUL alors écrire("Match nul")
Sinon écrire(le_joueur)
      écrire("a gagné. Félicitations!")
fin si
```

6)Programme ADA