

Projet Othello - LA CONCEPTION DÉTAILLÉE

Groupe 1.5

3 décembre 2015

Première partie

Conception détaillée des TAD

1 Conception détaillée des types

1.1 CD du type « Couleur »

— **Type** Couleur = {blanc, noir}

1.2 CD du type « Pion »

— **Type** Pion = Couleur

1.3 CD du type « Position »

— **Type** Position = **Structure**
 ligne : **Naturel**
 colonne : **Naturel**
finstructure

1.4 CD du type « Plateau »

— **Type** Position = **Structure**
 pions : **Tableau**[1..8][1..8] de Pion
 presencePions : **Tableau**[1..8][1..8] de **Booleen**
finstructure

1.5 CD du type « Coup »

— **Type** Coup = **Structure**
 position : Position
 pion : Pion
finstructure

1.6 CD du type « Coups »

— **Type** Coups = **Structure**
 tabCoups : **Tableau**[1..60] de Coup

```
    nbCps : Naturel  
finstructure
```

2 Conception détaillée des opérations des TAD

Deuxième partie

Conception détaillée des algorithmes compliqués de l'analyse « faireUnePartie »

1 La procédure « faireUnePartie »

procédure faireUnePartie (**E** afficher : afficherPlateau, obtenirCoupJoueur1, obtenirCoupJoueur2 : getCoup, **S** joueur : Couleur, estMatchNul : **Booleen**)

Déclaration plateau : Plateau
aPuJouerJoueur1, aPuJouerJoueur2, estFinie : **Booleen**
couleurJoueur1, couleurJoueur2 : Couleur
nbPionsBlancs, nbPionsNoirs : **Naturel**

debut

```
aPuJouerJoueur1 ← VRAI  
aPuJouerJoueur2 ← VRAI  
couleurJoueur1 ← blanc()  
couleurJoueur2 ← noir()  
estFinie ← FAUX  
nbPionsBlancs ← 2  
nbPionsNoirs ← 2  
plateau ← initialiserPlateau()  
afficher(plateau)  
tant que non(estFinie) faire  
    jouer(plateau, couleurJoueur1, obtenirCoupJoueur1, aPuJouerJoueur1)  
    afficher(plateau)  
    jouer(plateau, couleurJoueur2, obtenirCoupJoueur1, aPuJouerJoueur2)  
    afficher(plateau)  
    finPartie(aPuJouerJoueur1, aPuJouerJoueur2, plateau, estFinie, nbPionsBlancs, nbPionsNoirs)
```

fintantque

si nbPionsBlancs = nbPionsNoirs **alors**

```
    joueur ← blanc()  
    estMatchNul ← VRAI
```

sinon

```
    estMatchNul ← FAUX
```

si nbPionsBlancs > nbPionsNoirs **alors**

```
    joueur ← blanc()
```

sinon

```
    joueur ← noir()
```

finsi

finsi

fin

2 La procédure « jouer »

procédure jouer (**E/S** plateau : Plateau, couleurJoueur : Couleur, **E** obtenirCoupJoueur : getCoup, **S** aPuJouer : **Booleen**)

Déclaration i : **Naturel**

coups : Coups

joueurCourant : Couleur

coupJoueur : Coup

debut

coupJoueur \leftarrow obtenirCoupJoueur(plateau,couleurJoueur)

coups \leftarrow listeCoupsPossibles(plateau, couleurJoueur)

pour i \leftarrow 1 **à** nbCoups(coups) **faire**

si iemeCoup(coups,i) = coup **alors**

jouerCoup(coupJoueur,plateau)

finsi

finpour

aPuJouer \leftarrow res

fin

3 La procédure « jouerCoup »

procédure jouerCoup (**E** coup : Coup, **E/S** plateau : Plateau)

Déclaration i : **NaturelNonNul**

debut

poserPion(plateau, obtenirPositionCoup(coup), obtenirPionCoup(coup))

pos \leftarrow obtenirPositionCoup(coup)

pionJoueur \leftarrow obtenirPionCoup(coup)

inverserPions(pos, pionJoueur, plateau : Plateau)

fin

4 La procédure « inverserPions »

procédure inverserPions (**E** pos : Position, pionJoueur : Pion, **E/S** plateau : Plateau)

Déclaration posTmp : Position

x,y : **Entier**

i,j : **NaturelNonNul**

pionPresent : **Booleen**

debut

pour i \leftarrow 1 **à** 3 **faire**

x \leftarrow i - 2

pour j \leftarrow 1 **à** 3 **faire**

y \leftarrow i - 2

si non (x = 0) et (y = 0) **alors**

posTmp \leftarrow pos

pionEstPresent(pionJoueur, x, y, posTmp, plateau, pionPresent)

si pionPresent **alors**

inverserPionsDir(plateau, pos, posTmp, -x, -y, pionJoueur)

finsi

```

        finsi
    finpour
finpour
fin

```

5 La procédure « inverserPionsDir »

```

procédure inverserPionsDir (E/S plateau : Plateau, E posInitiale, posCourante : Pion, x, y : Entier)
debut
    si non (posInitiale = posCourante) alors
        inverserPion(plateau, posCourante)
        posCourante ← fixerPosition(x+i, y+j)
        inverserPionsDir(plateau, posInitiale, posCourante, x, y)
    finsi
fin

```

6 La procédure « pionEstPresent »

```

procédure pionEstPresent (E pionJoueur : Pion, x, y : Entier, E/S pos : Position, plateau : Plateau,
S pionPresent : Booleen)
    Déclaration i, j : NaturelNonNul
debut
    i ← obtenirLigne(pos)
    j ← obtenirColonne(pos)
    si ((x+i)<1) ou ((x+i)>8) ou ((y+j)<1) ou ((y+j)>8) alors
        pionPresent ← FAUX
    sinon
        pos ← fixerPosition(x+i, y+j)
        pionEstPresentRecuratif(pionJoueur, x, y, pos, plateau, pionPresent)
    finsi
fin

```

7 La procédure « pionEstPresentRecuratif »

```

procédure pionEstPresentRecuratif (E pionJoueur : Pion, x, y : Entier, E/S pos : Position, plateau :
Plateau, S pionPresent : Booleen)
    Déclaration i, j : NaturelNonNul
        couleurJoueur : Couleur
debut
    i ← obtenirLigne(pos)
    j ← obtenirColonne(pos)
    couleurJoueur ← obtenirCouleurPion(pionJoueur)
    si estCaseVide(plateau, pos) alors
        pionPresent ← FAUX
    sinon
        si obtenirCouleurPion(obtenirPion(plateau, pos)) = couleurJoueur alors
            pionPresent ← VRAI

```

```

    sinon
        si ((x+i)<1) ou ((x+i)>8) ou ((y+j)<1) ou ((y+j)>8) alors
            pionPresent ← FAUX
        sinon
            pos ← fixerPosition(x+i, y+j)
            pionEstPresentRecuratif(pionJoueur, x, y, pos, plateau, pionPresent)
        finsi
    finsi
finsi
fin

```

Troisième partie

Conception détaillée des algorithmes compliqués de l'analyse « obtenirCoupIA »

1 La fonction « obtenirCoupIA »

fonction obtenirCoupIA (plateau : Plateau, couleur : Couleur) : Coup

Déclaration i, profondeurMinMax : **Naturel**
 coupsPossibles : Coups
 plateauTest : Plateau
 scoreCourant, meilleurScore : **Entier**
 coupCourant, meilleurCoup : Coup

debut

```

    profondeurMinMax ← profondeur()
    coupsPossibles ← listeCoupsPossibles(plateau, couleur)
    si nbCoups(coupsPossibles) > 0 alors
        meilleurCoup ← iemeCoup(coupsPossibles, 1)
        plateauTest ← copierPlateau(plateau)
        meilleurScore ← scoreDUnCoup(plateauTest, meilleurCoup, couleur)
        pour i ← 2 à nbCoups(coupsPossibles) faire
            coupCourant ← iemeCoup(coupsPossibles, i)
            plateauTest ← copierPlateau(plateau)
            scoreCourant ← scoreDUnCoup(plateauTest, coupCourant, couleur)
            si scoreCourant > meilleurScore alors
                meilleurCoup ← coupCourant
                meilleurScore ← scoreCourant
        finsi
    finpour
finsi

```

```

    retourner meilleurCoup
fin

```

fin

2 La fonction « scoreDUnCoup »

fonction scoreDUnCoup (plateau : Plateau, coup : Coup, couleurRef, couleurCourante : Couleur, profondeurCourante : **Naturel**) : **Entier**

debut

 jouerCoup(coup, plateau)

si plateauRempli(plateau) ou profondeurCourante = 0 **alors**

retourner score(plateau, couleurRef)

sinon

retourner minMax(plateau, couleurRef, changerCouleur(CouleurCourante), profondeurCourante - 1)

finsi

fin

3 La fonction « coupValide »

fonction coupValide (plateau : Plateau, coup : Coup) : **Booleen**

Déclaration pos, posTmp : Position

 pionJoueur : Pion

 pionPresent : **Booleen**

 x, y : **Entier**

debut

 x ← -1

 pionPresent ← FAUX

 pos ← obtenirPositionCoup(coup)

 pionJoueur ← obtenirPionCoup(coup)

tant que non(pionPresent) et (x < 2) **faire**

 y ← -1

tant que non(pionPresent) et (y < 2) **faire**

si non((x = 0) et (y = 0)) **alors**

 posTmp ← pos

 pionEstPresent(pionJoueur, x, y, posTmp, plateau, pionPresent)

si pionPresent **alors**

si (|obtenirLigne(posTmp) - obtenirLigne(pos)| < 2) ou (|obtenirColonne(posTmp) - obtenirColonne(pos)| < 2) **alors**

 pionPresent ← FAUX

finsi

finsi

finsi

 y ← y + 1

fintantque

 x ← x + 1

fintantque

retourner pionPresent

fin

4 La fonction « minMax »

fonction minMax (plateau : Plateau, couleurRef, couleurCourante : Couleur, profondeurCourante : Naturel) : Entier

Déclaration coupsPossibles : Coups
 resultat, score : **Entier**
 i : **Naturel**

debut

 coupsPossibles \leftarrow listeCoupsPossibles(plateau, couleurCourante)

si nbCoups(coupsPossibles) > 0 **alors**

 resultat \leftarrow scoreDUnCoup(plateau, iemeCoup(coupsPossibles, 1), couleurRef, couleurCourante, profondeurCourante)

pour i \leftarrow 2 **à** nbCoups(coupsPossibles) **faire**

 score \leftarrow scoreDUnCoup(plateau, iemeCoup(coupsPossibles, i), couleurRef, couleurCourante, profondeurCourante)

si couleurCourante = couleurRef **alors**

 resultat \leftarrow max(resultat, score)

sinon

 resultat \leftarrow min(resultat, score)

finsi

finpour

sinon

si couleurCourante = couleurRef **alors**

 resultat \leftarrow INFINI

sinon

 resultat \leftarrow - INFINI

finsi

finsi

retourner resultat

fin

***Remarque :** On utilise ici une constante « **INFINI** », qui représentera un score supérieur à tout autre score, c'est-à-dire un coup gagnant.*