

Projet Othello - LA CONCEPTION DÉTAILLÉE

Groupe 1.5

26 novembre 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 = **Tableau**[1..8] de Coup

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, plateauModifie : **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, plateauModifie)
afficher(plateau)
jouer(plateau, couleurJoueur2, obtenirCoupJoueur1, aPuJouerJoueur2, plateauModifie)
afficher(plateau)
finPartie(aPuJouerJoueur1, aPuJouerJoueur2, plateau, estFinie, nbPionsBlancs, nbPionsNoirs)

fin tant que

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

res : **Booleen**

debut

coupJoueur ← obtenirCoupJoueur(plateau,couleurJoueur)

coups ← listeCoupsPossibles(plateau, couleurJoueur)

pour i ← 1 **à** nbCoups(coups) **faire**

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

 res ← VRAI

 jouerCoup(coupJoueur,plateau)

sinon

 res ← FAUX

finsi

finpour

aPuJouer ← res

fin

3 La procédure « jouerCoup »

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

Déclaration i : **NaturelNonNul**

aModifie,plateauDifferent : **Booleen**

pas : **Entier**

debut

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

plateauDifferent ← FAUX

pour i ← 1 **à** 3 **pas de 2 faire**

 pas ← i − 2

 inverserLigne(aModifie, pas, obtenirPositionCoup(coup), obtenirPionCoup(coup), plateau)

si aModifie **alors**

 plateauDifferent ← VRAI

finsi

 inverserColonne(aModifie, pas, obtenirPositionCoup(coup), obtenirPionCoup(coup), plateau)

si aModifie **alors**

 plateauDifferent ← VRAI

finsi

 inverserDiagMontante(aModifie, pas, obtenirPositionCoup(coup), obtenirPionCoup(coup), plateau)

si aModifie **alors**

 plateauDifferent ← VRAI

finsi

 inverserDiagDescendante(aModifie, pas, obtenirPositionCoup(coup), obtenirPionCoup(coup), plateau)

```

    si aModifie alors
        plateauDifferent  $\leftarrow$  VRAI
    finsi
finpour
fin

```

4 La procédure « inverserLigne »

procédure inverserLigne (**E** pas : **Entier**, pos : Position, pionJoueur : Pion, **E/S** plateau : Plateau)

Déclaration i,j,k,l : **Naturel**
test : **Booleen**

```

debut
    i  $\leftarrow$  obtenirLigne(pos)
    j  $\leftarrow$  obtenirColonne(pos)
    k  $\leftarrow$  j + pas
    test  $\leftarrow$  FAUX
    tant que ((k > 0) et (k  $\leq$  8) et (test = FAUX) et non(estCaseVide(plateau, fixerPosition(i,k))))
    faire
        si obtenirPion(plateau, fixerPosition(i, k)) = pionJoueur alors
            test  $\leftarrow$  VRAI
        sinon
            k  $\leftarrow$  k + pas
        finsi
    fintantque
    si test alors
        pour l  $\leftarrow$  k - pas à j + pas faire
            inverserPion(plateau, fixerPosition(i, l))
        finpour
    finsi
fin

```

5 La procédure « inverserColonne »

procédure inverserColonne (**E** pas : **Entier**, pos : Position, pionJoueur : Pion, **E/S** plateau : Plateau, **S** test : **Booleen**)

Déclaration i, j, k, l : **Naturel**
test : **Booleen**

```

debut
    i  $\leftarrow$  obtenirLigne(pos)
    j  $\leftarrow$  obtenirColonne(pos)
    k  $\leftarrow$  j + pas
    test  $\leftarrow$  FAUX
    tant que ((k > 0) et (k  $\leq$  8) et (test = FAUX) et non(estCaseVide(plateau, fixerPosition(k,j))))
    faire
        si obtenirPion(plateau, fixerPosition(k,j)) = pionJoueur alors
            test  $\leftarrow$  VRAI
        sinon
            k  $\leftarrow$  k + pas
    fin

```

```

    finsi
fintantque
si test alors
    pour l  $\leftarrow$  k - pas à i + pas faire
        inverserPion(plateau, fixerPosition(l,j))
    finpour
finsi
fin

```

6 La procédure « inverserDiagMontante »

procédure inverserDiagMontante (**E** pas : **Entier**, pos : Position, pionJoueur : Pion, **E/S** plateau : Plateau)

Déclaration i, j, k, l, m, n : **Naturel**

```

debut
    i  $\leftarrow$  obtenirLigne(pos)
    j  $\leftarrow$  obtenirColone(pos)
    k  $\leftarrow$  j + pas
    m  $\leftarrow$  i - 1
    test  $\leftarrow$  FAUX
    tant que ((k > 0) et (k  $\leq$  8) et (m > 0) et (test = FAUX) et non(estCaseVide(plateau, fixerPosition(m, k)))) faire
        si obtenirPion(plateau, fixerPosition(m, k)) = pionJoueur alors
            test  $\leftarrow$  VRAI
        sinon
            k  $\leftarrow$  j + pas
            m  $\leftarrow$  m - 1
        finsi
    fintantque
    si test alors
        l  $\leftarrow$  m + 1
        n  $\leftarrow$  k - pas
        repeter
            inverserPion(plateau, fixerPosition(l, n))
            l  $\leftarrow$  l + 1
            n  $\leftarrow$  n - pas
        jusqu'a ce que l = i
    finsi
fin

```

7 La procédure « inverserDiagDescendante »

procédure inverserDiagDescendante (**E** pas : **Entier**, pos : Position, pionJoueur : Pion, **E/S** plateau : Plateau)

Déclaration i, j, k, l, m, n : **Naturel**

```

debut
    i  $\leftarrow$  obtenirLigne(pos)
    j  $\leftarrow$  obtenirColone(pos)

```

```

k ← j + pas
m ← i + 1
test ← FAUX
tant que ((k > 0) et (k ≤ 8) et (m ≤ 8) et (test = FAUX) et non(estCaseVide(plateau, fixerPosition(m, k)))) faire
    si obtenirPion(plateau, fixerPosition(m, k)) = pionJoueur alors
        test ← VRAI
    sinon
        k ← j + pas
        m ← m + 1
    fin
fin
si test alors
    l ← m - 1
    n ← k - pas
    repeter
        inverserPion(plateau, fixerPosition(l, n))
        l ← l - 1
        n ← n - pas
    jusqu'a ce que l = i
fin
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

```

2 La fonction « coupValide »

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

Déclaration estValide, aVoisinAutreCouleur : **Booleen**

pos : Position

plateauTmp : Plateau

i,j : **Naturel**

couleurJoueur, autreCouleur : Couleur

debut

couleurJoueur ← obtenirCouleurPion(obtenirPionCoup(coup))

autreCouleur ← changerCouleur(couleurJoueur)

pos ← obtenirPositionCoup(coup)

i ← obtenirLigne(pos)

j ← obtenirColonne(pos)

aVoisinAutreCouleur ← obtenirCouleurPion(obtenirPion(plateau,fixerPosition(i-1,j-1)))=autreCouleur
ou

obtenirCouleurPion(obtenirPion(plateau,fixerPosition(i-1,j)))=autreCouleur ou

obtenirCouleurPion(obtenirPion(plateau,fixerPosition(i-1,j+1)))=autreCouleur ou

obtenirCouleurPion(obtenirPion(plateau,fixerPosition(i,j-1)))=autreCouleur ou

obtenirCouleurPion(obtenirPion(plateau,fixerPosition(i,j+1)))=autreCouleur ou

obtenirCouleurPion(obtenirPion(plateau,fixerPosition(i+1,j-1)))=autreCouleur ou

obtenirCouleurPion(obtenirPion(plateau,fixerPosition(i+1,j)))=autreCouleur ou

obtenirCouleurPion(obtenirPion(plateau,fixerPosition(i+1,j+1)))=autreCouleur

si aVoisinAutreCouleur **alors**

plateauTmp ← copierPlateau(plateau, plateauTmp

jouerCoup(coup, plateauTmp, estValide

finsi

retourner estValide

fin