Projet Othello - La conception détaillée

Groupe 1.5

25 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 = **Structure**
  couleur : Couleur
finstructure

1.3 CD du type « `Position` »

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

1.4 CD du type « `Plateau` »

— **Type** Case = {blanc, noir, vide}
— **Type** Plateau = **Tableau[**1..8**][**1..8**] de** Case

1.5 CD du type « `Coup` »

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

1.6 CD du type « `Coups` »

— **Type** Coups = **Structure**
  listeDesCoups : Liste<Coup>

finstructure

2 Conception détaillée des opérations des TAD

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

1 La procédure « jouer »

procédure jouer (**E/S** plateau : Plateau, **E** coupJouer : getCoup, **S** aPuJouer : **Booleen**)

Déclaration i : **Naturel**
coups : Coups
joueurCourant : Couleur
res : **Booleen**

debut
joueurCourant ← obtenirCouleurPion(obtenirPionCoup(coupJoueur))
coups ← listeCoupsPossibles(plateau, joueurCourant)
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

2 La procédure « jouerCoup »

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

Déclaration i : **NaturelNonNul**
pas : **Entier**

debut
poserPion(plateau, obtenirPositionCoup(coup), obtenirPionCoup(coup))
pour i ←1 **à** 3 **pas de** 2 **faire**
pas ← i − 2
inverserLigne(pas, obtenirPositionCoup(coup), obtenirPionCoup(coup), plateau)
inverserColonne(pas, obtenirPositionCoup(coup), obtenirPionCoup(coup), plateau)
inverserDiagMontante(pas, obtenirPositionCoup(coup), obtenirPionCoup(coup), plateau)
inverserDiagDescendante(pas, obtenirPositionCoup(coup), obtenirPionCoup(coup), plateau)
finpour
fin

3 La procédure « inverserColonne »

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

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

debut
i ← obtenirLigne(pos)
j ← obtenirColonne(pos)
k ← 2 + pas
test ← FAUX
tant que ((k > 0) et (k ⩽ 8) et (test = FAUX) et non(estCaseVide(plateau, fixerPosition(k,j))))
faire
si obtenirPion(plateau, fixerPosition(k,j)) = pionJoueur **alors**
test ← VRAI
sinon
k ← k + pas
finsi
fintantque
si test **alors**
pour l ←k − pas **à** i + pas **faire**
inverserPion(plateau, fixerPosition(l,j))
finpour
finsi
fin

4 La procédure « faireUnePartie »

procédure faireUnePartie (**E** afficher : afficherPlateau, coupJoueur1, coupJoueur2 : getCoup, **S** joueur : Couleur, estMatchNul : **Booleen**)

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

debut
aPuJouerJoueur1 ← VRAI
aPuJouerJoueur2 ← VRAI
estFinie ← VRAI
nbPionsBlancs ← 2
nbPionsNoirs ← 2
plateau ← initialiserPlateau()
finPartie(aPuJouerJoueur1, aPuJouerJoueur2, plateau, estFinie, nbPionsBlancs, nbPionsNoirs)
afficher(plateau)
tant que non(estFinie) **faire**
jouer(coupJoueur1, plateau, aPuJouerJoueur1)
afficher(plateau)
jouer(coupJoueur2, plateau, 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
```

Troisième partie

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

1 La fonction « `obtenirCoupIA` »

```
fonction obtenirCoupIA (plateau : Plateau, couleur : Couleur) : Coup
    Déclaration  i, pronfondeurMinMax : Naturel
                 coupsPossibles : Coups
                 scoreCourant, meilleurScore : Entier
                 coupCourant, meilleurCoup : Coup
debut
    profondeurMinMax ← profondeur()
    coupsPossibles ← listeCoupsPossibles(plateau,couleur)
    si nbCoups(coupsPossibles) > 0 alors
        meilleurCoup ← iemeCoup(coupsPossibles,1)
        meilleurScore ← scoreDUnCoup(plateau,meilleurCoup,couleur)
        pour i ←2 à nbCoups(coupsPossibles faire
            coupCourant ← iemeCoup(coupsPossibles,i)
            scoreCourant ← scoreDUnCoup(plateau,coupCourant,couleur)
            si scoreCourant > meilleurScore alors
                meilleurCoup ← coupCourant
                meilleurScore ← scoreCourant
            finsi
        finpour
    finsi
fin
```