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 « `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

2 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 ← VRAI
      finsi
    finpour
fin
```

3 La procédure « `inverserColonne` »

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

```
  Déclaration  i, j, k, l : NaturelNonNul
               test : Booleen
debut
  i ← obtenirLigne(pos)
  j ← obtenirColonne(pos)
  k ← j + 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, 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)

   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
```