

# Projet Othello - LA CONCEPTION DÉTAILLÉE

Groupe 1.5

1<sup>er</sup> 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 = **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 : **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  
res : **Booleen**

**debut**

coupsJoueur ← obtenirCoupJoueur(plateau,couleurJoueur)

coups ← listeCoupsPossibles(plateau, couleurJoueur)

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

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

        jouerCoup(coupJoueur,plateau,res)

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

**si** aModifie **alors**

        plateauDifferent ← VRAI

**finsi**

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

**si** aModifie **alors**

        plateauDifferent ← VRAI

**finsi**

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

**si** aModifie **alors**

        plateauDifferent ← VRAI

**finsi**

**finpour**

**fin**

## 4 La procédure « inverserLigne »

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

**Déclaration** i,j,k,l : **Naturel**  
                  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(i,k))))

**faire**

**si** obtenirPion(plateau, fixerPosition(i, k)) = pionJoueur **alors**

            test ← VRAI

**si** (k > j+1) ou (k < j-1) **alors**

            aModifie ← VRAI

**finsi**

**sinon**

            k ← k + pas

**finsi**

**fintantque**

**si** test **alors**

**pour** l ← 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** aModifie : **Booleen**)

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

**debut**

    i ← obtenirLigne(pos)

    j ← obtenirColonne(pos)

    k ← i + 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

**si** (k > i+1) ou (k < i-1) **alors**

            aModifie ← VRAI

**finsi**

**sinon**

            k ← k + pas

```

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

```

## 6 La procédure « inverserDiag »

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

**Déclaration**  $i, j, k, l, m, n, o$  : **Naturel**  
pasDiag : **Entier**

```

debut
     $i \leftarrow \text{obtenirLigne}(\text{pos})$ 
     $j \leftarrow \text{obtenirColonne}(\text{pos})$ 
     $k \leftarrow j + \text{pas}$ 
    test  $\leftarrow$  FAUX
    pour  $o \leftarrow 1$  à 3 pas de 2 faire
        pasDiag  $\leftarrow o - 2$ 
         $m \leftarrow i + \text{pasDiag}$ 
        tant que  $((k > 0) \text{ et } (k \leq 8) \text{ et } (m > 0) \text{ et } (\text{test} = \text{FAUX}) \text{ et } \text{non}(\text{estCaseVide}(\text{plateau}, \text{fixerPosition}(m, k))))$  faire
            si obtenirPion(plateau, fixerPosition(m, k)) = pionJoueur alors
                test  $\leftarrow$  VRAI
                si  $(k > j + 1) \text{ ou } (k < j - 1) \text{ ou } (m > i + 1) \text{ ou } (m < i - 1)$  alors
                    aModifie  $\leftarrow$  VRAI
                finsi
            sinon
                 $k \leftarrow j + \text{pas}$ 
                 $m \leftarrow m + \text{pasDiag}$ 
            finsi
        fintantque
    si test alors
         $l \leftarrow m - \text{pasDiag}$ 
         $n \leftarrow k - \text{pas}$ 
        repeter
            inverserPion(plateau, fixerPosition(l, n))
             $l \leftarrow l - \text{pasDiag}$ 
             $n \leftarrow n - \text{pas}$ 
        jusqu'a ce que  $(l = i) \text{ et } (n = j)$ 
    finsi
finpour
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  $\leftarrow$  profondeur()  
coupsPossibles  $\leftarrow$  listeCoupsPossibles(plateau, couleur)  
**si** nbCoups(coupsPossibles) > 0 **alors**  
    meilleurCoup  $\leftarrow$  iemeCoup(coupsPossibles, 1)  
    plateauTest  $\leftarrow$  copierPlateau(plateau)  
    meilleurScore  $\leftarrow$  scoreDUnCoup(plateauTest, meilleurCoup, couleur)  
    **pour** i  $\leftarrow$  2 **à** nbCoups(coupsPossibles) **faire**  
        coupCourant  $\leftarrow$  iemeCoup(coupsPossibles, i)  
        plateauTest  $\leftarrow$  copierPlateau(plateau)  
        scoreCourant  $\leftarrow$  scoreDUnCoup(plateauTest, coupCourant, couleur)  
        **si** scoreCourant > meilleurScore **alors**  
            meilleurCoup  $\leftarrow$  coupCourant  
            meilleurScore  $\leftarrow$  scoreCourant  
    **finsi**  
    **finpour**  
    **finsi**  
    **retourner** meilleurCoup  
**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** 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**

### 4 La fonction « minMax »

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

**Déclaration** coupsPossibles : Coups

resultat, score : **Entier**

i : **Naturel**

**debut**

coupsPossibles ← listeCoupsPossibles(plateau, couleurCourante)

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

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

**pour** i ← 2 **à** nbCoups(coupsPossibles) **faire**

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

**si** couleurCourante = couleurRef **alors**

resultat ← 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.*