

# INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE ROUEN

EC ALGORITHMIQUE AVANCÉE ET PROGRAMMATION C

---

## Rapport de projet d'algorithmique

---

*Titre du projet :*

« Jeu d'Othello »

*Auteurs :*

Gautier DARCHEN

Romain JUDIC

Riadh KILANI

Claire LOVISA

Sandratra RASENDRASOA

6 décembre 2015



# Introduction

# Table des matières

<b>Introduction</b>	<b>2</b>
<b>I Analyse</b>	<b>4</b>
<b>1 Analyse des TAD</b>	<b>5</b>
1.1 Le TAD « Couleur » . . . . .	5
1.2 Le TAD « Plateau » . . . . .	5
1.3 Le TAD « Coup » . . . . .	5
1.4 Le TAD « Pion » . . . . .	6
1.5 Le TAD « Coups » . . . . .	6
1.6 Le TAD « Position » . . . . .	6
<b>2 Analyse descendante</b>	<b>7</b>
<b>II Conception préliminaire</b>	<b>8</b>
<b>1 Conception préliminaire des TAD</b>	<b>9</b>
1.1 Conception préliminaire du TAD « Couleur » . . . . .	9
1.2 Conception préliminaire du TAD « Plateau » . . . . .	9
1.3 Conception préliminaire du TAD « Coup » . . . . .	9
1.4 Conception préliminaire du TAD « Pion » . . . . .	9
1.5 Conception préliminaire du TAD « Coups » . . . . .	10
1.6 Conception préliminaire du TAD « Position » . . . . .	10
<b>2 Conception préliminaire des fonctions et procédures issues des analyses descendantes</b>	<b>11</b>
2.1 Conception préliminaire de l'analyse descendante de « Faire une partie » . . . . .	11
2.1.1 Types . . . . .	11
2.1.2 Sous-programmes . . . . .	11
2.2 Conception préliminaire de l'analyse descendante de « obtenirCoupIA » . . . . .	11
<b>III Conception détaillée</b>	<b>13</b>
<b>1 Conception détaillée des TAD</b>	<b>14</b>
1.1 CD du type « Couleur » . . . . .	14
1.2 CD du type « Pion » . . . . .	14
1.3 CD du type « Position » . . . . .	14
1.4 CD du type « Plateau » . . . . .	14
1.5 CD du type « Coup » . . . . .	14

1.6	CD du type « Coups » . . . . .	14
<b>2</b>	<b>Conception détaillée des algorithmes compliqués de l'analyse « faireUnePartie »</b>	<b>15</b>
2.1	La procédure « faireUnePartie » . . . . .	15
2.2	La procédure « jouer » . . . . .	16
2.3	La procédure « jouerCoup » . . . . .	16
2.4	La procédure « inverserPions » . . . . .	16
2.5	La procédure « inverserPionsDir » . . . . .	17
2.6	La procédure « pionEstPresent » . . . . .	17
2.7	La procédure « pionEstPresentRecuratif » . . . . .	17
<b>3</b>	<b>Conception détaillée des algorithmes compliqués de l'analyse « obtenirCoupIA »</b>	<b>19</b>
3.1	La fonction « obtenirCoupIA » . . . . .	19
3.2	La fonction « scoreDUnCoup » . . . . .	19
3.3	La fonction « coupValide » . . . . .	20
3.4	La fonction « minMax » . . . . .	20
<b>IV</b>	<b>Développement</b>	<b>22</b>
	<b>Conclusion</b>	<b>23</b>

Première partie

*Analyse*

# Chapitre 1

## Analyse des TAD

### 1.1 Le TAD « Couleur »

**Nom:** Couleur  
**Opérations:** blanc:  $\rightarrow$  Couleur  
noir:  $\rightarrow$  Couleur  
changerCouleur: Couleur  $\rightarrow$  Couleur  
**Axiomes:** - *changerCouleur(blanc())=noir()*  
- *changerCouleur(noir())=blanc()*

### 1.2 Le TAD « Plateau »

**Nom:** Plateau  
**Utilise:** **Booleen**, Position, Pion  
**Opérations:** creerPlateau:  $\rightarrow$  Plateau  
estCaseVide: Plateau  $\times$  Position  $\rightarrow$  **Booleen**  
viderCase: Plateau  $\times$  Position  $\rightarrow$  Plateau  
poserPion: Plateau  $\times$  Position  $\times$  Pion  $\rightarrow$  Plateau  
obtenirPion: Plateau  $\times$  Position  $\rightarrow$  Pion  
inverserPion: Plateau  $\times$  Position  $\rightarrow$  Plateau  
**Axiomes:** - *estCaseVide(creerPlateau(),position)=VRAI*  
- *estCaseVide(viderCase(plateau,position),position)=VRAI*  
- *estCaseVide(poserPion(plateau,position,pion),position)=FAUX*  
- *obtenirPion(poserPion(plateau,position,pion),position)=pion*  
- *inverserPion(inverserPion(plateau,position),position)=plateau*  
**Préconditions:** viderCase(plateau,position): non(estCaseVide(plateau,position))  
poserPion(plateau,position): estCaseVide(plateau,position)  
obtenirPion(plateau,position): non(estCaseVide(plateau,position))  
inverserPion(plateau,position): non(estCaseVide(plateau,position))

### 1.3 Le TAD « Coup »

**Nom:** Coup  
**Utilise:** Position, Pion  
**Opérations:** creerCoup: Position  $\times$  Pion  $\rightarrow$  Coup

obtenirPositionCoup:  $\text{Coup} \rightarrow \text{Position}$

obtenirPionCoup:  $\text{Coup} \rightarrow \text{Pion}$

**Axiomes:**

- $\text{obtenirPositionCoup}(\text{creerCoup}(\text{pos}, \text{pion})) = \text{pos}$
- $\text{obtenirPionCoup}(\text{creerCoup}(\text{pos}, \text{pion})) = \text{pion}$

## 1.4 Le TAD « Pion »

**Nom:** Pion

**Utilise:** Couleur

**Opérations:**

- creerPion:  $\text{Couleur} \rightarrow \text{Pion}$
- obtenirCouleurPion:  $\text{Pion} \rightarrow \text{Couleur}$
- retournerPion:  $\text{Pion} \rightarrow \text{Pion}$

**Axiomes:**

- $\text{obtenirCouleurPion}(\text{creerPion}(\text{couleur})) = \text{couleur}$
- $\text{obtenirCouleurPion}(\text{retournerPion}(\text{pion})) = \text{changerCouleur}(\text{obtenirCouleurPion}(\text{pion}))$

## 1.5 Le TAD « Coups »

**Nom:** Coups

**Utilise:** Naturel, NaturelNonNul, Coup

**Opérations:**

- creerCoups:  $\rightarrow \text{Coups}$
- ajouterCoups:  $\text{Coups} \times \text{Coup} \rightarrow \text{Coups}$
- nbCoups:  $\text{Coups} \rightarrow \text{Naturel}$
- iemeCoup:  $\text{Coups} \times \text{NaturelNonNul} \rightarrow \text{Coup}$

**Axiomes:**

- $\text{iemeCoup}(\text{ajouterCoups}(\text{cps}, \text{cp}), \text{nbCoups}(\text{cps})) = \text{cp}$
- $\text{nbCoups}(\text{creerCoups}()) = 0$
- $\text{nbCoups}(\text{ajouterCoups}(\text{cps}, \text{cp})) = \text{nbCoups}(\text{cps}) + 1$

**Préconditions:**  $\text{iemeCoup}(\text{cps}, i): i \leq \text{nbCoups}(\text{cps})$

## 1.6 Le TAD « Position »

**Nom:** Position

**Utilise:** NaturelNonNul

**Opérations:**

- obtenirLigne:  $\text{Position} \rightarrow \text{NaturelNonNul}$
- obtenirColonne:  $\text{Position} \rightarrow \text{NaturelNonNul}$
- fixerPosition:  $\text{NaturelNonNul} \times \text{NaturelNonNul} \rightarrow \text{Position}$

**Axiomes:**

- $\text{obtenirLigne}(\text{fixerPosition}(\text{ligne}, \text{colonne})) = \text{ligne}$
- $\text{obtenirColonne}(\text{fixerPosition}(\text{ligne}, \text{colonne})) = \text{colonne}$

**Préconditions:**  $\text{fixerPosition}(\text{ligne}, \text{colonne}): 1 \leq \text{ligne} \leq 8 \ \& \ 1 \leq \text{colonne} \leq 8$



## Chapitre 2

# Analyse descendante

On insérera ici les images des analyses descendantes (une fois qu'elles seront finies et qu'on n'y touchera plus).

Deuxième partie

Conception préliminaire

# Chapitre 1

## Conception préliminaire des TAD

### 1.1 Conception préliminaire du TAD « Couleur »

- **fonction** blanc () : Couleur
- **fonction** noir () : Couleur
- **fonction** changerCouleur (couleur : Couleur) : Couleur

Pour la conception détaillée, nous avons ajouté la fonction de comparaison de deux « Couleur » :

- **fonction** sontEgales (couleur1, couleur2 : Couleur) : **Booleen**

### 1.2 Conception préliminaire du TAD « Plateau »

- **fonction** creerPlateau () : Plateau
- **fonction** estCaseVide (plateau : Plateau, position : Position) : Couleur
- **procédure** viderCase (**E/S** plateau : Plateau, **E** position : Position)  
  |**précondition(s)** non(estCaseVide(plateau,position))
- **procédure** poserPion (**E/S** plateau : Plateau, **E** position : Position, pion : Pion)  
  |**précondition(s)** estCaseVide(plateau,position)
- **fonction** obtenirPion (plateau : Plateau, position : Position) : Pion  
  |**précondition(s)** non(estCaseVide(plateau,position))
- **procédure** inverserPion (**E/S** plateau : Plateau, **E** position : Position)  
  |**précondition(s)** non(estCaseVide(plateau,position))

### 1.3 Conception préliminaire du TAD « Coup »

- **fonction** creerCoup (position : Position, pion : Pion) : Coup
- **fonction** obtenirPositionCoup (coup : Coup) : Position
- **fonction** obtenirPionCoup (coup : Coup) : Pion

Pour la conception détaillée, nous avons ajouté la fonction de comparaison de deux « Coup » :

- **fonction** sontEgaux (coup1, coup2 : Coup) : **Booleen**

### 1.4 Conception préliminaire du TAD « Pion »

- **fonction** creerPion (couleur : Couleur) : Pion

- **fonction** obtenirCouleurPion (pion : Pion) : Couleur
- **procédure** retournerPion (**E/S** pion : Pion)

Pour la conception détaillée, nous avons ajouté la fonction de comparaison de deux « Pion » :

- **fonction** sontEgaux (pion1, pion2 : Pion) : **Booleen**

## 1.5 Conception préliminaire du TAD « Coups »

- **fonction** creerCoups () : Coups
- **procédure** ajouterCoups (**E/S** coups : Coups, **E** coup : Coup)
- **fonction** nbCoups (coups : Coups) : Naturel
- **fonction** iemeCoup (coups : Coups, i : **NaturelNonNul**) : Coup  
  |précondition(s)  $i \leq \text{nbCoups}(\text{coups})$

## 1.6 Conception préliminaire du TAD « Position »

- **fonction** obtenirLigne (position : Position) : **NaturelNonNul**
- **fonction** obtenirColonne (position : Position) : **NaturelNonNul**
- **procédure** fixerPosition (**E** ligne, colonne : **NaturelNonNul**, **S** position : Position)  
  |précondition(s)  $1 \leq \text{ligne} \leq 8 \ \& \ 1 \leq \text{colonne} \leq 8$

Pour la conception détaillée, nous avons ajouté la fonction de comparaison de deux « Position » :

- **fonction** sontEgales (position1, position2 : Position) : **Booleen**

## Chapitre 2

# Conception préliminaire des fonctions et procédures issues des analyses descendantes

### 2.1 Conception préliminaire de l'analyse descendante de « Faire une partie »

#### 2.1.1 Types

- **Type** getCoup = **fonction**(plateau : Plateau, pionJoueur : Pion) : Coup
- **Type** afficherPlateau = **procédure**(E plateau : Plateau)

#### 2.1.2 Sous-programmes

- **procédure** faireUnePartie (E coupJoueur1, coupJoueur2 : getCoup, afficher : afficherPlateau, S joueur : Couleur, estMatchNul : **Booleen**)
- **fonction** initialiserPlateau () : Plateau
- **procédure** jouer (E/S plateau : Plateau, couleurJoueur : Couleur, E coupJoueur : getCoup, S aPuJouer : **Booleen**)
- **procédure** finPartie (E aPuJouerJoueur1, aPuJouerJoueur2 : **Booleen**, plateau : Plateau, S estFinie : **Booleen**, nbPionsBlancs, nbPionsNoirs : **Naturel**)
- **fonction** plateauRempli (plateau : Plateau) : **Booleen**
- **procédure** nbPions (E plateau : Plateau, S nbPionsBlancs, nbPionsNoirs : **Naturel**)
- **procédure** jouerCoup (E coup : Coup, E/S plateau : Plateau)
- **procédure** inverserPions (E pos : Position, pionJoueur : Pion, E/S plateau : Plateau)
- **procédure** inverserPionsDir (E/S plateau : Plateau, E posInitiale, posCourante : Position, x, y : Entier)
- **procédure** pionEstPresent (E pionJoueur : Pion, x, y : Entier, E/S pos : Position, plateau : Plateau, S pionPresent : **Booleen**)
- **procédure** pionEstPresentRecuratif (E pionJoueur : Pion, x, y : Entier, E/S pos : Position, plateau : Plateau, S pionPresent : **Booleen**)

### 2.2 Conception préliminaire de l'analyse descendante de « obtenirCoupIA »

- **fonction** obtenirCoupIA (plateau : Plateau, couleur : Couleur) : Coup
- **fonction** profondeur () :

- **fonction** listeCoupsPossibles (plateau : Plateau, couleur : Couleur) : Coups
- **fonction** coupValide (plateau : Plateau, coup : Coup) : **Booleen**
- **procédure** copierPlateau (**E** plateauACopier : Plateau, **S** plateauCopie : Plateau)
- **fonction** minMax (plateau : Plateau, couleurRef, couleurCourante : Couleur, profondeurCourante : Naturel) : **Entier**
- **fonction** scoreDUnCoup (plateau : Plateau, coup : Coup, couleurRef, couleurCourante : Couleur, profondeurCourante : Naturel) : **Entier**
- **fonction** score (plateau : Plateau, couleur : Couleur) : **Entier**
- **fonction** evaluerPlateau (plateau : Plateau, couleur : Couleur) : **Entier**

Troisième partie

Conception détaillée

# Chapitre 1

## Conception détaillée des TAD

### 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**



## Chapitre 2

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

### 2.1 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)
    finPartie(aPuJouerJoueur1, aPuJouerJoueur2, plateau, estFinie, nbPionsBlancs, nbPionsNoirs)
    jouer(plateau, couleurJoueur2, obtenirCoupJoueur1, aPuJouerJoueur2)
    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
fin

```

## 2.2 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 ← obtenirCoupJoueur(plateau,couleurJoueur)
  coups ← listeCoupsPossibles(plateau, couleurJoueur)
  pour i ← 1 à nbCoups(coups) faire
    si iemeCoup(coups,i) = coup alors
      jouerCoup(coupJoueur,plateau)
    finsi
  finpour
  aPuJouer ← res
fin

```

## 2.3 La procédure « jouerCoup »

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

**Déclaration** i : **NaturelNonNul**

```

debut
  poserPion(plateau, obtenirPositionCoup(coup), obtenirPionCoup(coup))
  pos ← obtenirPositionCoup(coup)
  pionJoueur ← obtenirPionCoup(coup)
  inverserPions(pos, pionJoueur, plateau : Plateau)
fin

```

## 2.4 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 ← 1 à 3 faire
    x ← i - 2

```

```

    pour j ← 1 à 3 faire
        y ← i - 2
        si non (x = 0) et (y = 0) alors
            posTmp ← pos
            pionEstPresent(pionJoueur, x, y, posTmp, plateau, pionPresent)
            si pionPresent alors
                inverserPionsDir(plateau, pos, posTmp, -x, -y, pionJoueur)
            finsi
        finsi
    finpour
finpour
fin

```

## 2.5 La procédure « inverserPionsDir »

**procédure** inverserPionsDir (**E/S** plateau : Plateau, **E** posInitiale, posCourante : Position, 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**

## 2.6 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)  
 pionEstPresentRecursif(pionJoueur, x, y, pos, plateau, pionPresent)  
 finsi  
**fin**

## 2.7 La procédure « pionEstPresentRecursif »

**procédure** pionEstPresentRecursif (**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
```

## Chapitre 3

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

### 3.1 La fonction « obtenirCoupIA »

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

**Déclaration** i, profondeurMinMax : **Naturel**  
coupsPossibles : Coups  
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)  
    meilleurScore  $\leftarrow$  scoreDUnCoup(plateau, meilleurCoup, couleur)  
    **pour** i  $\leftarrow$  2 **à** nbCoups(coupsPossibles) **faire**  
        coupCourant  $\leftarrow$  iemeCoup(coupsPossibles, i)  
        scoreCourant  $\leftarrow$  scoreDUnCoup(plateau, coupCourant, couleur)  
        **si** scoreCourant > meilleurScore **alors**  
            meilleurCoup  $\leftarrow$  coupCourant  
            meilleurScore  $\leftarrow$  scoreCourant

**finsi**

**finpour**

**finsi**

**retourner** meilleurCoup

**fin**

### 3.2 La fonction « scoreDUnCoup »

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

**Déclaration** plateauTest : Plateau

**debut**

```

    plateauTest ← copierPlateau(plateau)
    jouerCoup(coup, plateauTest)
    si plateauRempli(plateauTest) ou profondeurCourante = 0 alors
        retourner score(plateauTest, couleurRef)
    sinon
        retourner minMax(plateauTest, couleurRef, changerCouleur(couleurCourante), profondeurCourante - 1)
    finsi

```

**fin**

### 3.3 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**

### 3.4 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.*

Quatrième partie

Développement



# Conclusion