

# 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

29 décembre 2015



# Introduction

# Table des matières

<b>Introduction</b>	<b>2</b>
<b>I Analyse</b>	<b>5</b>
<b>1 Analyse des TAD</b>	<b>6</b>
1.1 Le TAD « Couleur » . . . . .	6
1.2 Le TAD « Pion » . . . . .	6
1.3 Le TAD « Position » . . . . .	6
1.4 Le TAD « Plateau » . . . . .	6
1.5 Le TAD « Coup » . . . . .	7
1.6 Le TAD « Coups » . . . . .	7
<b>2 Analyse descendante</b>	<b>8</b>
<b>II Conception préliminaire</b>	<b>9</b>
<b>1 Conception préliminaire des TAD</b>	<b>10</b>
1.1 Conception préliminaire du TAD « Couleur » . . . . .	10
1.2 Conception préliminaire du TAD « Pion » . . . . .	10
1.3 Conception préliminaire du TAD « Position » . . . . .	10
1.4 Conception préliminaire du TAD « Plateau » . . . . .	11
1.5 Conception préliminaire du TAD « Coup » . . . . .	11
1.6 Conception préliminaire du TAD « Coups » . . . . .	11
<b>2 Conception préliminaire des fonctions et procédures issues des analyses descendantes</b>	<b>12</b>
2.1 Conception préliminaire de l'analyse descendante de « Faire une partie » . . . . .	12
2.1.1 Types . . . . .	12
2.1.2 Sous-programmes . . . . .	12
2.2 Conception préliminaire de l'analyse descendante de « obtenirCoupIA » . . . . .	12
<b>III Conception détaillée</b>	<b>14</b>
<b>1 Conception détaillée des TAD</b>	<b>15</b>
1.1 CD du type « Couleur » . . . . .	15
1.2 CD du type « Pion » . . . . .	15
1.3 CD du type « Position » . . . . .	15
1.4 CD du type « Plateau » . . . . .	15
1.5 CD du type « Coup » . . . . .	15

1.6	CD du type « Coups » . . . . .	15
<b>2</b>	<b>Conception détaillée des algorithmes compliqués de l'analyse « faireUnePartie »</b>	<b>17</b>
2.1	La procédure « faireUnePartie » . . . . .	17
2.2	La procédure « jouer » . . . . .	18
2.3	La procédure « jouerCoup » . . . . .	18
2.4	La procédure « inverserPions » . . . . .	18
2.5	La procédure « inverserPionsDir » . . . . .	19
2.6	La procédure « pionEstPresent » . . . . .	19
2.7	La procédure « pionEstPresentRecuratif » . . . . .	20
<b>3</b>	<b>Conception détaillée des algorithmes compliqués de l'analyse « obtenirCoupIA »</b>	<b>21</b>
3.1	La fonction « obtenirCoupIA » . . . . .	21
3.2	La fonction « scoreDUnCoup » . . . . .	21
3.3	La fonction « coupValide » . . . . .	22
3.4	La fonction « minMax » . . . . .	22
3.5	La fonction « evaluerPlateau » . . . . .	23
3.6	La fonction « evaluerNbCoupsPossiblesAdversaire » . . . . .	23
3.7	La fonction « evaluerNbPionsCouleur » . . . . .	24
3.8	La fonction « evaluerPositionsPionsPlateau » . . . . .	24
<b>IV</b>	<b>Développement</b>	<b>25</b>
<b>V</b>	<b>Répartition du travail</b>	<b>26</b>
<b>1</b>	<b>Analyse descendante</b>	<b>27</b>
<b>2</b>	<b>Conception préliminaire</b>	<b>28</b>
<b>3</b>	<b>Conception détaillée</b>	<b>29</b>
<b>4</b>	<b>Développement</b>	<b>30</b>
	<b>Conclusion</b>	<b>30</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 « Pion »

**Nom:** Pion  
**Utilise:** Couleur  
**Opérations:** creerPion: Couleur  $\rightarrow$  Pion  
obtenirCouleurPion: Pion  $\rightarrow$  Couleur  
retournerPion: Pion  $\rightarrow$  Pion  
**Axiomes:** - *obtenirCouleurPion(creerPion(couleur))=couleur*  
- *obtenirCouleurPion(retournerPion(pion))=changerCouleur(obtenirCouleurPion(pion))*

### 1.3 Le TAD « Position »

**Nom:** Position  
**Utilise:** NaturelNonNul  
**Opérations:** obtenirLigne: Position  $\rightarrow$  NaturelNonNul  
obtenirColonne: Position  $\rightarrow$  NaturelNonNul  
fixerPosition: NaturelNonNul  $\times$  NaturelNonNul  $\rightarrow$  Position  
**Axiomes:** - *obtenirLigne(fixerPosition(ligne,colonne))=ligne*  
- *obtenirColonne(fixerPosition(ligne,colonne))=colonne*  
**Préconditions:** fixerPosition(ligne,colonne):  $1 \leq \text{ligne} \leq 8 \ \& \ 1 \leq \text{colonne} \leq 8$

### 1.4 Le TAD « Plateau »

**Nom:** Plateau  
**Utilise:** Booleen, Position, Pion

**Opérations:**  $\text{creerPlateau} : \rightarrow \text{Plateau}$   
 $\text{estCaseVide} : \text{Plateau} \times \text{Position} \rightarrow \text{Booleen}$   
 $\text{viderCase} : \text{Plateau} \times \text{Position} \rightarrow \text{Plateau}$   
 $\text{poserPion} : \text{Plateau} \times \text{Position} \times \text{Pion} \rightarrow \text{Plateau}$   
 $\text{obtenirPion} : \text{Plateau} \times \text{Position} \rightarrow \text{Pion}$   
 $\text{inverserPion} : \text{Plateau} \times \text{Position} \rightarrow \text{Plateau}$

**Axiomes:** -  $\text{estCaseVide}(\text{creerPlateau}(), \text{position}) = \text{VRAI}$   
-  $\text{estCaseVide}(\text{viderCase}(\text{plateau}, \text{position}), \text{position}) = \text{VRAI}$   
-  $\text{estCaseVide}(\text{poserPion}(\text{plateau}, \text{position}, \text{pion}), \text{position}) = \text{FAUX}$   
-  $\text{obtenirPion}(\text{poserPion}(\text{plateau}, \text{position}, \text{pion}), \text{position}) = \text{pion}$   
-  $\text{inverserPion}(\text{inverserPion}(\text{plateau}, \text{position}), \text{position}) = \text{plateau}$

**Préconditions:**  $\text{viderCase}(\text{plateau}, \text{position}) : \neg \text{estCaseVide}(\text{plateau}, \text{position})$   
 $\text{poserPion}(\text{plateau}, \text{position}) : \text{estCaseVide}(\text{plateau}, \text{position})$   
 $\text{obtenirPion}(\text{plateau}, \text{position}) : \neg \text{estCaseVide}(\text{plateau}, \text{position})$   
 $\text{inverserPion}(\text{plateau}, \text{position}) : \neg \text{estCaseVide}(\text{plateau}, \text{position})$

## 1.5 Le TAD « Coup »

**Nom:** Coup  
**Utilise:** Position, Pion  
**Opérations:**  $\text{creerCoup} : \text{Position} \times \text{Pion} \rightarrow \text{Coup}$   
 $\text{obtenirPositionCoup} : \text{Coup} \rightarrow \text{Position}$   
 $\text{obtenirPionCoup} : \text{Coup} \rightarrow \text{Pion}$

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

## 1.6 Le TAD « Coups »

**Nom:** Coups  
**Utilise:** Naturel, NaturelNonNul, Coup  
**Opérations:**  $\text{creerCoups} : \rightarrow \text{Coups}$   
 $\text{ajouterCoups} : \text{Coups} \times \text{Coup} \rightarrow \text{Coups}$   
 $\text{nbCoups} : \text{Coups} \rightarrow \text{Naturel}$   
 $\text{iemeCoup} : \text{Coups} \times \text{NaturelNonNul} \rightarrow \text{Coup}$

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



## 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

Nous avons mis en place un code d'identification à l'aide de préfixes pour chaque TAD de la manière suivante :

- Couleur : CL\_\_
- Pion : PI\_\_
- Position : POS\_\_
- Plateau : PL\_\_
- Coup : CP\_\_
- Coups : CPS\_\_

### 1.1 Conception préliminaire du TAD « Couleur »

- **fonction** CL\_blanc () : Couleur
- **fonction** CL\_noir () : Couleur
- **fonction** CL\_changerCouleur (couleur : Couleur) : Couleur

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

- **fonction** CL\_sontEgales (couleur1, couleur2 : Couleur) : **Booleen**

### 1.2 Conception préliminaire du TAD « Pion »

- **fonction** PI\_creerPion (couleur : Couleur) : Pion
- **fonction** PI\_obtenirCouleurPion (pion : Pion) : Couleur
- **procédure** PI\_retournerPion (**E/S** pion : Pion)

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

- **fonction** PI\_sontEgaux (pion1, pion2 : Pion) : **Booleen**

### 1.3 Conception préliminaire du TAD « Position »

- **fonction** POS\_obtenirLigne (position : Position) : **NaturelNonNul**
- **fonction** POS\_obtenirColonne (position : Position) : **NaturelNonNul**
- **procédure** POS\_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** POS\_sontEgales (position1, position2 : Position) : **Booleen**

## 1.4 Conception préliminaire du TAD « Plateau »

- **fonction** PL\_creerPlateau () : Plateau
- **fonction** PL\_estCaseVide (plateau : Plateau, position : Position) : Couleur
- **procédure** PL\_viderCase (**E/S** plateau : Plateau, **E** position : Position)  
|**précondition**(s) non(estCaseVide(plateau,position))
- **procédure** PL\_poserPion (**E/S** plateau : Plateau, **E** position : Position, pion : Pion)  
|**précondition**(s) estCaseVide(plateau,position)
- **fonction** PL\_obtenirPion (plateau : Plateau, position : Position) : Pion  
|**précondition**(s) non(estCaseVide(plateau,position))
- **procédure** PL\_inverserPion (**E/S** plateau : Plateau, **E** position : Position)  
|**précondition**(s) non(estCaseVide(plateau,position))

## 1.5 Conception préliminaire du TAD « Coup »

- **fonction** CP\_creerCoup (position : Position, pion : Pion) : Coup
- **fonction** CP\_obtenirPositionCoup (coup : Coup) : Position
- **fonction** CP\_obtenirPionCoup (coup : Coup) : Pion

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

- **fonction** CP\_sontEgaux (coup1, coup2 : Coup) : **Booleen**

## 1.6 Conception préliminaire du TAD « Coups »

- **fonction** CPS\_creerCoups () : Coups
- **procédure** CPS\_ajouterCoups (**E/S** coups : Coups, **E** coup : Coup)
- **fonction** CPS\_nbCoups (coups : Coups) : Naturel
- **fonction** CPS\_iemeCoup (coups : Coups, i : **NaturelNonNul**) : Coup  
|**précondition**(s)  $i \leq \text{nbCoups}(\text{coups})$

## 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**)
- **procédure** initialiserPlateau (E/S plateau : 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 () : **NaturelNonNul**

- **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 = **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** 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] deCoup



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)
            finsi
        finsi
    finpour
fin

```

## 2.5 La procédure « inverserPionsDir »

**procédure** inverserPionsDir (**E/S** plateau : Plateau, **E** posInitiale, posCourante : Position, x, y : **Entier**)

**Déclaration** i,j : NaturelNonNul

**debut**

```

    i ← obtenirLigne(posCourante)
    j ← obtenirColonne(posCourante)
    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  
couleurAdversaire : Couleur

**debut**

```

    i ← obtenirLigne(pos)
    j ← obtenirColonne(pos)
    couleurAdversaire ← changerCouleur(obtenirCouleur(pionJoueur))
    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)
        si sontEgales(obtenirCouleur(obtenirPion(plateau, pos)), couleurAdversaire) alors
            pos ← fixerPosition(2×x+i, 2×y+j)
            pionEstPresentRecursif(pionJoueur, x, y, pos, plateau, pionPresent)
        sinon
            pionPresent ← FAUX
    finsi

```

**finsi**  
**fin**

## 2.7 La procédure « pionEstPresentRecuratif »

**procédure** pionEstPresentRecuratif (**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 ← profondeur()  
coupsPossibles ← listeCoupsPossibles(plateau, couleur)  
**si** nbCoups(coupsPossibles) > 0 **alors**  
    meilleurCoup ← iemeCoup(coupsPossibles, 1)  
    meilleurScore ← scoreDUnCoup(plateau, meilleurCoup, couleur, couleur, profondeurMinMax)  
    **pour** i ← 2 **à** nbCoups(coupsPossibles) **faire**  
        coupCourant ← iemeCoup(coupsPossibles, i)  
        scoreCourant ← scoreDUnCoup(plateau, coupCourant, couleur, couleur, profondeurMinMax)  
        **si** scoreCourant > meilleurScore **alors**  
            meilleurCoup ← coupCourant  
            meilleurScore ← 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)
    fin

```

**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)
            fin
            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 ← listeCoupsPossibles(plateau, couleurCourante)
    si nbCoups(coupsPossibles) > 0 alors
        resultat ← scoreDUnCoup(plateau, iemeCoup(coupsPossibles, 1), couleurRef, couleurCourante, profondeurCourante)
    fin

```

```

    pour i ← 2 à nbCoups(coupsPossibles) faire
        score ← scoreDUnCoup(plateau, iemeCoup(coupsPossibles, i), couleurRef, couleurCourante, pro-
            fondeurCourante)
        si couleurCourante = couleurRef alors
            resultat ← max(resultat, score)
        sinon
            resultat ← min(resultat, score)
        finsi
    finpour
    sinon
        si couleurCourante = couleurRef alors
            resultat ← INFINI
        sinon
            resultat ← - INFINI
        finsi
    fin
    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.*

### 3.5 La fonction « evaluerPlateau »

**fonction** evaluerPlateau (plateau : Plateau, couleur : Couleur) : **Entier**

**Déclaration** evaluer1, evaluer2, evaluer3, res : **Entier**

**debut**

```

    evaluer1 ← evaluerNbCoupsPossiblesAdversaire(plateau, couleur)
    evaluer2 ← evaluerNbPionsCouleur(plateau, couleur)
    evaluer3 ← evaluerPositionsPionsPlateau(plateau, couleur)
    res ← evaluer1 + evaluer2 + evaluer3
    retourner res
    
```

**fin**

### 3.6 La fonction « evaluerNbCoupsPossiblesAdversaire »

**fonction** evaluerNbCoupsPossiblesAdversaire (plateau : Plateau, couleur : Couleur) : **Entier**

**Déclaration** nbCoupsAdversaire, res : **Entier**

coupsAdversaire : Coups

couleurAdversaire : Couleur

**debut**

```

    couleurAdversaire ← changerCouleur(couleur)
    coupsAdversaire ← listeCoupsPossibles(plateau, couleurAdversaire)
    nbCoupsAdversaire ← nbCoups(coupsAdversaire)
    res ← 60 - 10 × nbCoupsAdversaire
    retourner res
    
```

**fin**



### 3.7 La fonction « evaluerNbPionsCouleur »

**fonction** evaluerNbPionsCouleur (plateau : Plateau, couleur : Couleur) : **Entier**

**Déclaration** res : **Entier**  
 nbPionsNoirs, nbPionsBlancs : **Naturel**

**debut**

nbPions(plateau,nbPionsNoirs,nbPionsBlancs)

**si** sontEgales(couleur,noir()) **alors**

res  $\leftarrow$  nbPionsNoirs−nbPionsBlancs

**sinon**

res  $\leftarrow$  nbPionsBlancs−nbPionsNoirs

**finsi**

**retourner** res

**fin**

### 3.8 La fonction « evaluerPositionsPionsPlateau »

**fonction** evaluerPositionsPionsPlateau (plateau : Plateau, couleur : Couleur) : **Entier**

**Déclaration** res, resJoueur, resAdversaire : **Entier**  
 i, j, x, y : **Naturel**  
 pos : Position  
 grilleScore : **Tableau**[1..8][1..8] **de Entier**

**debut**

grilleScore  $\leftarrow$  initialiserGrilleScore()

resJoueur  $\leftarrow$  0

resAdversaire  $\leftarrow$  0

**pour** i  $\leftarrow$  1 à 8 **faire**

**pour** j  $\leftarrow$  1 à 8 **faire**

fixerPosition(i−1,j−1,pos)

**si** non estCaseVide(plateau, pos) et sontEgales(obtenirCouleur(obtenirPion(pos)), couleur) **alors**

resJoueur  $\leftarrow$  resJoueur+grilleScore[i−1][j−1]

**sinon**

**si** non estCaseVide(plateau,pos) **alors**

resAdversaire  $\leftarrow$  resAdversaire+grilleScore[i−1][j−1]

**finsi**

**finsi**

**finpour**

**finpour**

**retourner** res

**fin**

Quatrième partie

Développement

Cinquième partie

Répartition du travail

# Chapitre 1

## Analyse descendante

Responsables	Claire	Riadh	Sandratra	Gautier	Romain
Sous-programme					
faireUnePartie					
initialiserPlateau					
jouer					
finPartie					
plateauRempli					
nbPions					
jouerCoup					
inverserPions					
inverserPionsDir					
pionEstPresent					
pionEstPresentRecuratif					
obtenirCoupIA					
profondeur					
listeCoupsPossibles					
coupValide					
copierPlateau					
minMax					
scoreDUnCoup					
score					
evaluerPlateau					

TABLE 1.1 – Répartition des tâches dans la phase d'analyse descendante

## Chapitre 2

# Conception préliminaire

Responsables		Claire	Riadh	Sandratra	Gautier	Romain
Sous-programme						
faireUnePartie						
initialiserPlateau						
jouer						
finPartie						
plateauRempli						
nbPions						
jouerCoup						
inverserPions						
inverserPionsDir						
pionEstPresent						
pionEstPresentRecuratif						
obtenirCoupIA						
profondeur						
listeCoupsPossibles						
coupValide						
copierPlateau						
minMax						
scoreDUnCoup						
score						
evaluerPlateau						
Type afficherPlateau						
Type getCoup						

TABLE 2.1 – Répartition des tâches dans la phase de conception préliminaire

## Chapitre 3

# Conception détaillée

## Chapitre 4

# Développement

	Fonction en C	Test unitaire associé
<code>afficher</code>	Gautier	
<code>TAD Couleur, Coups, Coup</code>	Gautier	Romain
<code>TAD Pion, Position, Plateau</code>	Claire	Romain
<code>faireUnePartie</code>	Riadh	
<code>initialiserPlateau</code>	Riadh, Gautier	Claire
<code>jouer</code>	Riadh	
<code>finPartie</code>	Riadh	Gautier
<code>plateauRempli</code>	Gautier	Claire
<code>nbPions</code>	Riadh, Gautier	Claire, Gautier
<code>jouerCoup</code>	Riadh	Claire
<code>inverserPion</code>	Riadh, Romain	Sandratra
<code>inverserPionDir</code>	Riadh, Romain	Sandratra
<code>pionEstPresent</code>	Riadh, Romain	Sandratra
<code>pionEstPresentRecuratif</code>	Riadh, Romain	
<code>obtenirCoupHumain</code>	Claire	Sandratra
<code>obtenirCoupIA</code>	Romain	Riadh
<code>profondeur</code>	Romain	
<code>listeCoupsPossibles</code>	Sandratra	Claire
<code>coupValide</code>	Sandratra	Claire
<code>copierPlateau</code>	Gautier	Claire
<code>minMax</code>	Gautier	
<code>scoreDUnCoup</code>	Gautier	?
<code>score</code>	Romain	?
<code>evaluerPlateau</code>	Claire	?
<code>main</code>	Gautier	

TABLE 4.1 – Répartition des tâches dans la phase de développement

# Conclusion