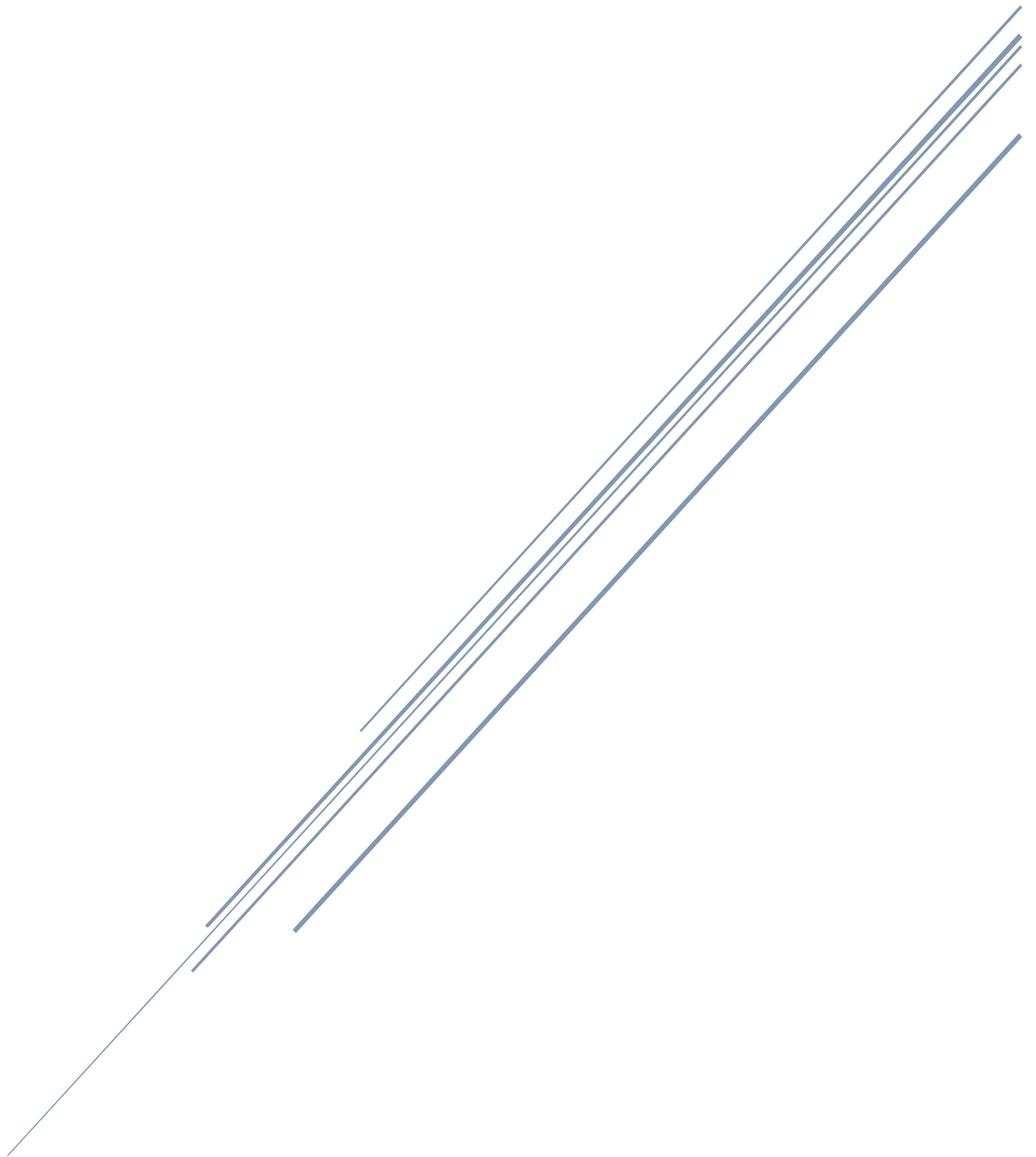


# ANALYSE JEU DU PUISSANCE 4

Thomas Vandewattyne & Legras Bryan  
Valentin Norro & Théo Cramez

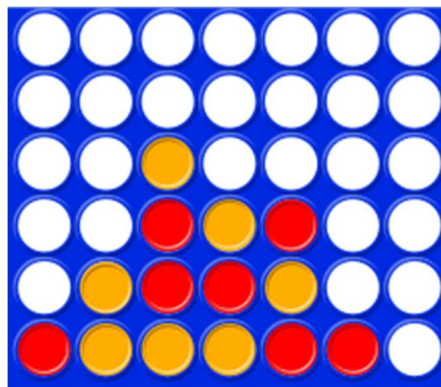


Algorithmie  
IUT Amiens – LP RGI (Option Web)

# Le Jeu du Puissance 4

Le jeu du puissance 4 est un jeu de stratégie combinatoire abstrait commercialisé pour la première fois en 1974 par la Milton Bradley Company.

Le but du jeu est d'aligner une suite de 4 pions de même couleur sur une grille comptant 6 rangées et 7 colonnes. Chaque joueur dispose de 21 pions d'une couleur. Tour à tour, les deux joueurs placent un pion dans la colonne de leur choix, le pion coulisse alors jusqu'à la position la plus basse possible dans ladite colonne à la suite de quoi c'est à l'adversaire de jouer. Le vainqueur est le joueur qui réalise le premier alignement (horizontal ou vertical ou diagonal) consécutif d'au moins quatre pions de sa couleur. Si, alors que toutes les cases de la grille de jeu sont remplies, aucun des deux joueurs n'a réalisé un tel alignement, la partie est déclarée nulle.



Lien vers le git : <https://github.com/Tvpirates/puissance-4>

Vidéo victoire en colonne :

<http://images.xelyos.fr/git/puissance-4/victoire-colonne.gif>

Vidéo victoire en ligne :

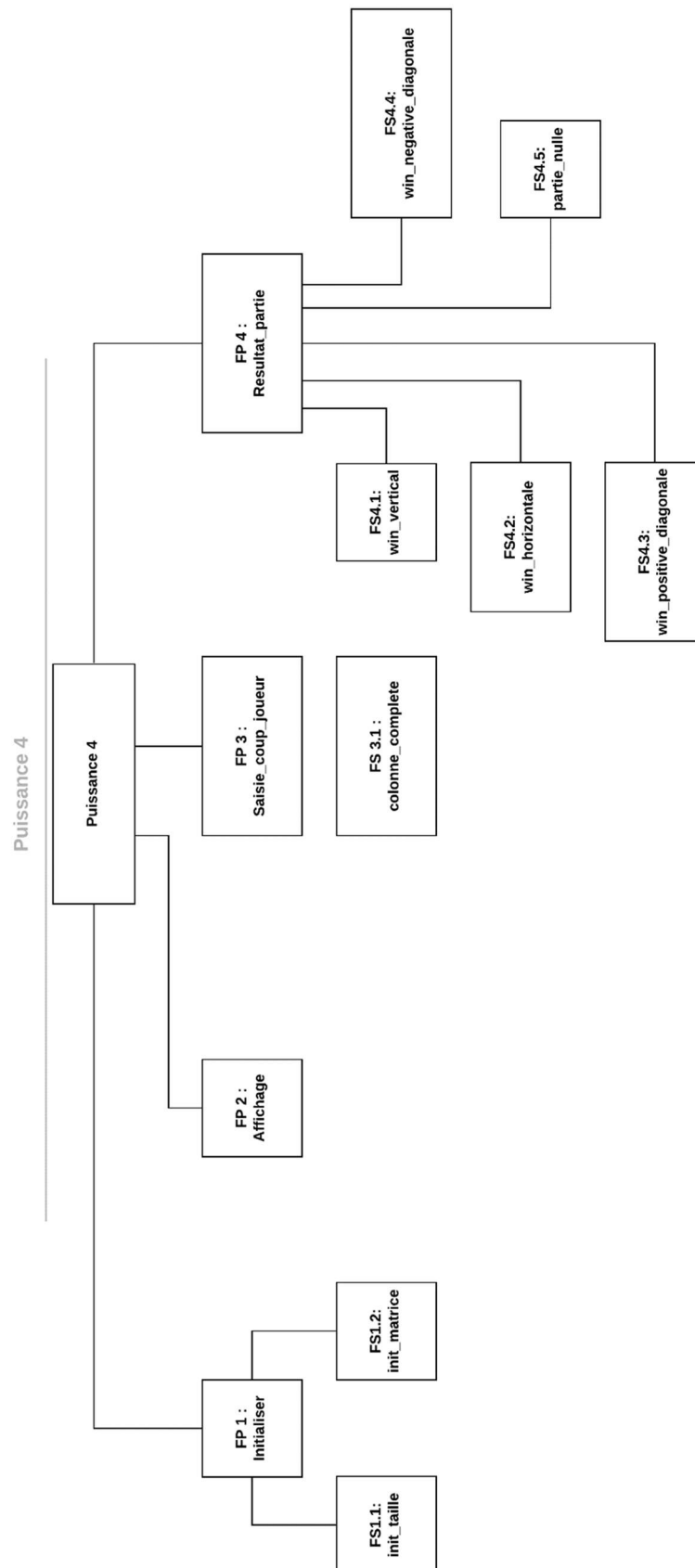
<http://images.xelyos.fr/git/puissance-4/victoire-ligne.gif>

Vidéo victoire en diagonale :

<http://images.xelyos.fr/git/puissance-4/victoire-diagonale.gif>

Vidéo partie nulle : <http://images.xelyos.fr/git/puissance-4/partie-nulle.gif>

# Analyse descendante du puissance 4



## FP1 : Initialiser

---

% Initialisation de la matrice de jeu

%IN []

%OUT [matrice]



- Dans un premier temps on définit la taille de la grille de jeu **FS1.1**
- Ensuite, on crée la matrice de jeu en fonction de la taille définie **FS1.2**
- Lorsque la grille est créée, on passe à l’affichage **FP2**

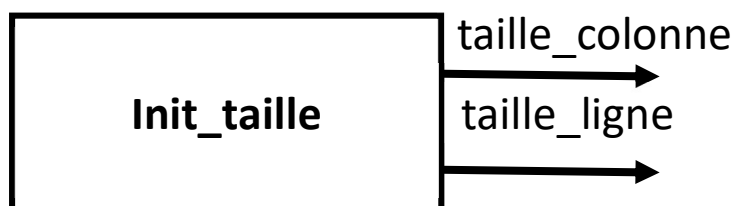
## FS1.1 : init\_taille

---

% Définition de la taille de la grille de jeu

%IN []

%OUT [taille\_colonne, taille\_ligne]



- On définit la taille en fonction de deux paramètres, le nombre de colonnes et le nombre de lignes

```
{
  "column": 7,
  "row": 6
}
```

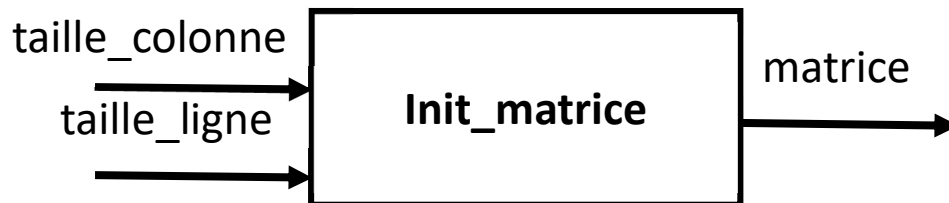
## FS1.2 : init\_matrice

---

% Création de la matrice de jeu

%IN [taille\_colonne, taille\_ligne]

%OUT [matrice]



- Nous allons initialiser des tableaux dans un tableau permettant ainsi d'avoir un tableau en 2 dimensions

Début

```
matrice ← tableau[taille_colonne]
Pour i allant de 0 à taille_colonne faire
    tableau_temporaire ← tableau[taille_ligne]
    Pour j allant de 0 à taille_ligne faire
        Tableau_temporaire[j] ← 0
    Fin pour
    matrice[i] ← tableau_temporaire
Fin pour
```

Fin

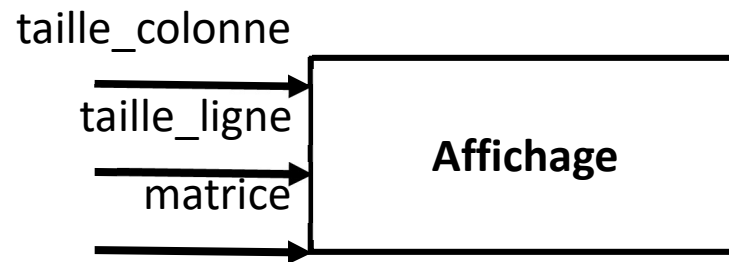
## FP2 : Affichage

---

% Affichage du tableau avec la position des pions

%IN [taille\_colonne, taille\_ligne, matrice]

%OUT []



- Nous allons afficher le tableau au joueur, en remplaçant les nombres (indiquant à qui appartient le pion) par des ronds de couleur
- Le jeton blanc ( $\bigcirc$ ) sera pour indiquer qu'aucun pion n'est placé à cet endroit
- Le jeton bleu ( $\odot$ ) pour le joueur 1
- Le jeton rouge ( $\bullet$ ) pour le joueur 2 ou l'IA

Début

```
jetons  $\leftarrow$  [ $\bigcirc$ ,  $\odot$ ,  $\bullet$ ]  
Pour c allant de 0 à taille_colonne faire  
    Pour l allant de 0 à taille_ligne faire  
        valeur  $\leftarrow$  matrice[c, l]  
        afficher jetons[valeur]  
    Fin pour  
Fin pour
```

Fin

### FP3 : Saisie\_coup\_joueur

---

% Saisir la position du jeton souhaitée par le joueur

% IN [taille\_colonne, taille\_ligne]

% OUT [position]



- Le joueur va choisir une colonne où il souhaite placer son jeton
- La colonne doit être comprise dans la zone de jeu
- S'il n'y a plus de place pour poser un jeton alors il doit ressaisir son coup

#### **FS3.1**

- Pour détecter si une colonne est pleine, il suffit de compter le nombre de zéros dans la colonne, si ce nombre est à 0, alors c'est qu'il n'y a plus de place pour jouer

Début

valide ← Faux

Tant que valide est *Faux* faire

    Afficher(Choisir une colonne entre 1 et *taille\_colonne*)

*position* ← Saisie du joueur

    Si *position* est compris entre 1 et *taille\_colonne* faire

        Si colonne\_complete(*position*) est *Faux* faire // FS3.1

            valide ← Vrai

        Fin si

    Fin si

Fin tant que

Fin

### FS3.1 : colonne\_complete

---

% On regarde que la colonne sélectionnée n'est pas pleine

%IN [position, matrice]

% OUT [estValide]



- Pour savoir si le coup est valide, nous devons regarder dans la colonne s'il y a encore des 0.
- S'il reste des 0 c'est que la colonne n'est pas complètement vide, on renvoie Vrai car on peut y placer le jeton
- Sinon c'est que la colonne est complète on renvoie donc Faux

Début

estValide  $\leftarrow$  Faux

Pour chaque valeur  $v$  de  $matrice[position]$  faire

Si  $v$  est égale à 0 faire

estValide  $\leftarrow$  Vrai

Fin si

Fin pour

Fin



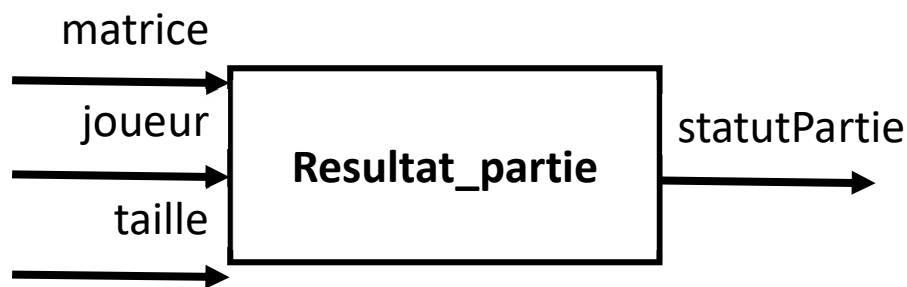
## FP4 : Resultat\_partie

---

% Vérification du statut de la partie

% IN [matrice, joueur, taille]

% OUT [statutPartie]



- On regarde l'alignement verticale **FS4.1**
- On regarde l'alignement horizontale **FS4.2**
- On regarde l'alignement diagonal positif **FS4.3**
- On regarde l'alignement diagonal négatif **FS4.4**
- On regarde si la partie est nulle **FS4.5**

Début

```
deplacement ← tableau[0, 1, 2, 3]
statutPartie ← nulle
statutPartie ← win_verticale // FS4.1
statutPartie ← win_horizontale // FS4.2
statutPartie ← win_positive_diagonale // FS4.3
statutPartie ← win_negative_diagonale // FS4.4
statutPartie ← partie_nulle //FS4.5
```

```
Si statutPartie est égale à Vrai alors
    Afficher(Partie Gagné)
Sinon si statutPartie est égale à Faux alors
    Afficher(Partie nulle)
```

Fin

## FS4.1 : win\_verticale

% La partie est gagnée par un joueur ayant ses pions alignés verticalement

%IN [matrice, joueur, taille, valeur]

% OUT [statutPartie]



- Nous allons regarder pour chaque colonne si les pions qui sont les uns à côté des autres appartiennent aux mêmes joueurs

Début

```
Pour column allant de 0 à taille_colonne faire
  Pour row allant de 0 à taille_ligne faire
    temporaire ← Vrai
    Pour v allant de 0 à taille(valeur) faire
      a ← valeur[v]
      Si (row+a) < taille_ligne faire
        Si matrice[column][row+a] != joueur faire
          temporaire ← Faux
        Fin si
      Fin si
    Fin pour
    Si temporaire est égale à Vrai alors
      statutPartie ← Vrai
    Fin si
  Fin pour
Fin pour
```

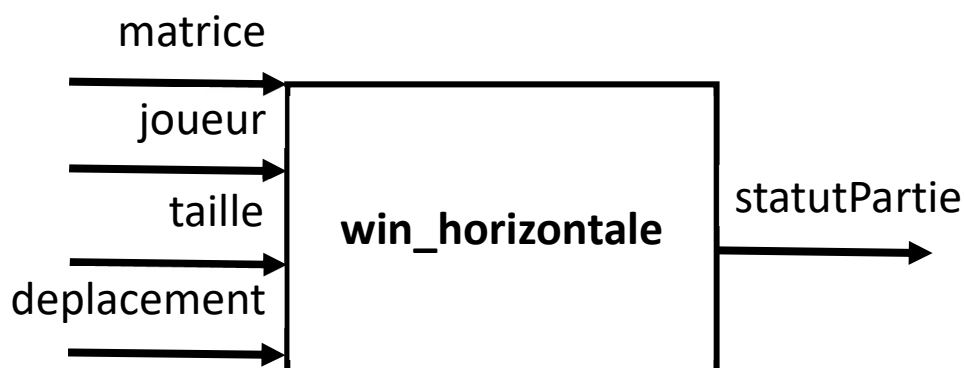
Fin

## FS4.2 : win\_horizontale

% La partie est gagnée par un joueur ayant ses pions alignés horizontalement

%IN [matrice, joueur, taille, valeur]

% OUT [statutPartie]



- Nous allons regarder pour chaque ligne si les pions qui sont les uns à côté des autres appartiennent aux mêmes joueurs

Début

```

Pour column allant de 0 à taille_colonne faire
  Pour row allant de 0 à taille_ligne faire
    temporaire ← Vrai
    Pour v allant de 0 à taille(valeur) faire
      a ← valeur[v]
      Si (column+a) < taille_colonne faire
        Si matrice[column+a][row] != joueur faire
          temporaire ← Faux
        Fin si
      Fin si
    Fin pour
    Si temporaire est égale à Vrai alors
      statutPartie ← Vrai
    Fin si
  Fin pour
Fin pour

```

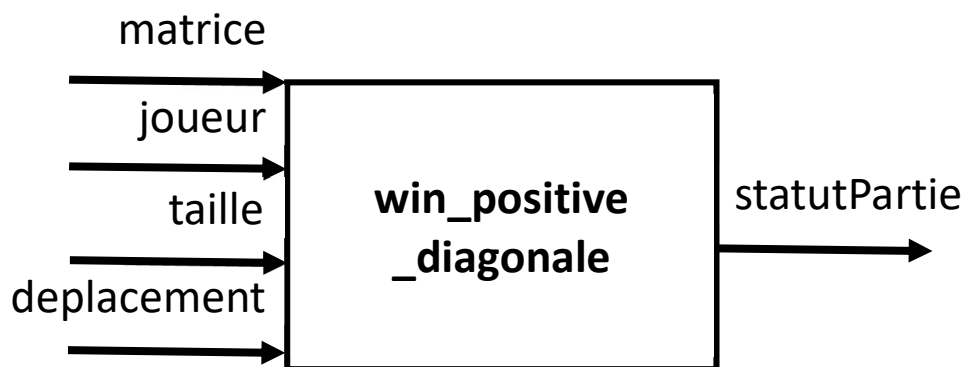
Fin

### FS4.3 : win\_positive\_diagonale

% La partie est gagnée par un joueur ayant ses pions alignés en diagonale positive

%IN [matrice, joueur, taille, valeur]

% OUT [statutPartie]



- Nous allons regarder pour chaque diagonale positive si les pions qui sont les uns à côté des autres appartiennent aux mêmes joueurs

Début

```
Pour column allant de 0 à taille_colonne faire
  Pour row allant de 0 à taille_ligne faire
    temporaire ← Vrai
    Pour v allant de 0 à taille(valeur) faire
      a ← valeur[v]
      Si (column+a) < taille_colonne et (row+a) < taille_ligne
        faire
          Si matrice[column+a][row+a] != joueur faire
            temporaire ← Faux
          Fin si
        Fin si
      Fin pour
    Si temporaire est égale à Vrai alors
      statutPartie ← Vrai
    Fin si
  Fin pour
Fin pour
```

Fin

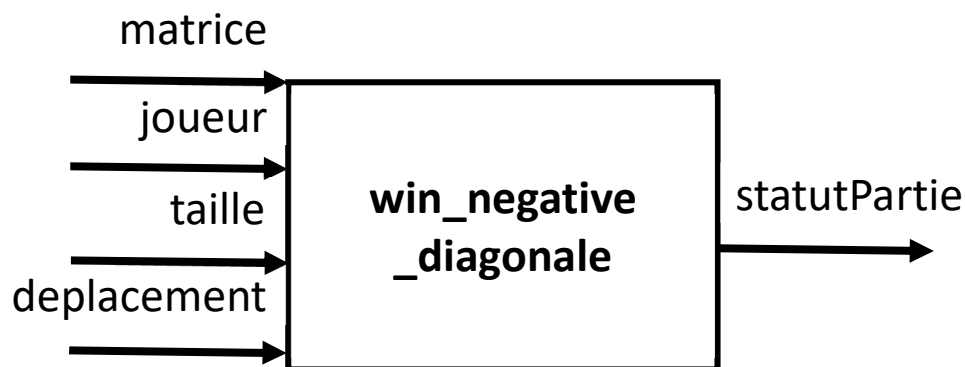
#### FS4.4 : win\_negative\_diagonale

---

% La partie est gagnée par un joueur ayant ses pions alignés en diagonale négative

%IN [matrice, joueur, taille, valeur]

% OUT [statutPartie]



- Nous allons regarder pour chaque diagonale négative si les pions qui sont les uns à côté des autres appartiennent aux mêmes joueurs

Début

```
Pour column allant de 0 à taille_colonne faire
  Pour row allant de 0 à taille_ligne faire
    temporaire ← Vrai
    Pour v allant de 0 à taille(valeur) faire
      a ← valeur[v]
      Si (column+a) < taille_colonne et (row+a) < taille_ligne
        faire
          Si matrice[column-a][row+a] != joueur faire
            temporaire ← Faux
          Fin si
        Fin si
      Fin pour
    Si temporaire est égale à Vrai alors
      statutPartie ← Vrai
    Fin si
  Fin pour
Fin pour
```

Fin

### FS4.5 : partie\_nulle

---

% Toutes les cases sont complétées par un jeton mais personne n'a gagné

% IN [matrice, taille\_colonne]

%OUT [statutPartie]



- Jusqu'ici, la partie n'est pas gagnée par l'un des joueurs
- Nous allons regarder s'il reste des cases disponibles pour poser un jeton
- S'il en reste alors la partie continue
- Sinon la partie est nulle

Début

*compteur*  $\leftarrow 0$

Pour *column* allant de 0 à *taille\_colonne* faire

    Si *colonne\_complete(column)* est *Vrai* faire

*compteur*  $\leftarrow$  *compteur* + 1

    Fin si

Fin pour

Si *compteur* est égale à *taille\_colonne* faire

*statutPartie*  $\leftarrow$  Faux

Fin si

Fin