Documentation expliquant les programmes Snort et Dodgem

Sommaire :

[1. Le programme SNORT 3](#_Toc181218305)

[a. Lancement du jeu 3](#_Toc181218306)

[b. La fonction snort(n) 3](#_Toc181218307)

[c. La fonction newBoard(n) 4](#_Toc181218308)

[d. La fonction displayBoard(board, n) 4](#_Toc181218309)

[e. La fonction selectSquare(board, n, player) 5](#_Toc181218310)

[f. La fonction possibleSquare(board, n, player, i, j) 6](#_Toc181218311)

[g. La fonction updateBoard(board, player, i, j) 6](#_Toc181218312)

[h. La fonction again(board, n, player) 7](#_Toc181218313)

[2. Le programme Dodgem 8](#_Toc181218314)

[a. La fonction Dodgem(n) 8](#_Toc181218315)

[b. La fonction newBoard(n) 9](#_Toc181218316)

[c. La fonction displayBoard(board, n) 10](#_Toc181218317)

[d. La fonction saisie\_taille\_plateau() 11](#_Toc181218318)

[e. La fonction saisie\_position(utilisateur, limite, type\_position) 11](#_Toc181218319)

[f. La fonction possiblePawn(board, n, directions, player, i, j) 12](#_Toc181218320)

[g. La fonction selectPawn(board, n, directions, player) 13](#_Toc181218321)

[h. La fonction possibleMove(board, n, directions, player, i, j, m) 14](#_Toc181218322)

[i. La fonction selectMove(board,n,directions,player,i,j) 15](#_Toc181218323)

[j. La fonction move(board, n, directions, player, i, j, m) 16](#_Toc181218324)

[k. La fonction bloquer(board, n, directions, player) 17](#_Toc181218325)

[l. La fonction win(board, n, directions, player) 17](#_Toc181218326)

# Le programme SNORT

## Lancement du jeu

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Ce code demande à l’utilisateur de saisir une taille de plateau.
* Un contrôle est réalisé pour s’assurer que la saisie est un entier positif.
* La fonction snort est lancée avec la taille du plateau spécifiée.

## La fonction snort(n)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* La fonction snort gère le déroulement du jeu entre deux joueurs.
* Elle initialise le plateau et le joueur 1 comme premier joueur.
* Tant qu'il reste des coups possibles, on affiche le plateau et on demande au joueur en cours de sélectionner une case, on met à jour le plateau, puis on passe au joueur suivant.
* Une fois la partie terminée, on annonce le gagnant et on affiche le plateau final.

## La fonction newBoard(n)

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

* Cette fonction initialise un plateau de jeu de taille n x n avec toutes les cases vides. Ce plateau est un tableau de n lignes et n colonnes.
* Crée une liste contenant n sous-listes, chacune remplie de 0, représentant des cases vides.
* Enfin on retourne le plateau créé.

## La fonction displayBoard(board, n)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Fonction qui affiche le plateau de jeu dans la console en formatant les lignes et les colonnes.
* On parcourt chaque ligne et colonne du plateau, affichant un . pour les cases vides (0), X pour le joueur 1 (1), et o pour le joueur 2 (2).
* On affiche une ligne de séparation sous le plateau et une ligne avec les numéros de colonne pour faciliter la lecture.

## La fonction selectSquare(board, n, player)

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

* Fonction qui demande au joueur de saisir des coordonnées valides pour poser un pion.
* On affiche un message pour le tour du joueur
* Demande une saisie pour la ligne et vérifie si elle est valide (un entier entre 1 et n). Si non valide, redemande à l’utilisateur.
* Même vérification pour la colonne.
* Si la case est jouable, retourne les coordonnées. Sinon, affiche un message d’erreur et redemande à l’utilisateur.

## La fonction possibleSquare(board, n, player, i, j)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Fonction qui vérifie si la case (i,j) est jouable pour le joueur, c'est-à-dire qu'elle est vide et non adjacente à une case contenant un pion adverse
* Si la case contient déjà un pion (différent de 0), la fonction retourne False (la case n’est pas jouable).
* On détermine l'adversaire en fonction du joueur actuel.
* On vérifie si une des cases adjacentes à la case (i,j) (en haut, en bas, à gauche, à droite) contient un pion adverse. Si oui, la fonction retourne False. Sinon, retourne True (la case est jouable).

## La fonction updateBoard(board, player, i, j)

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

* Fonction qui met à jour le plateau avec le coup du joueur.
* On place le pion du joueur (1 ou 2) dans la case (i, j) du plateau

## La fonction again(board, n, player)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Fonction qui vérifie s’il reste des coups possibles pour un joueur
* Parcourt chaque case du plateau et vérifie si elle est jouable pour ce joueur. Si oui, retourne True. Sinon, retourne False (fin du jeu pour ce joueur).

# Le programme Dodgem

## La fonction Dodgem(n)

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Cette fonction est la fonction principale pour exécuter le jeu du Dodgem.
* On définit les directions possibles pour le joueur 1 et pour le joueur 2.
* On crée un nouveau plateau de jeu de taille n x n.
* On définit le joueur 1 comme le 1er joueur à commencer la partie.
* La boucle principale du jeu qui est « While True » qui continue jusqu'à ce qu'une condition de victoire soit atteinte.
* Avec les if win, on vérifie si c’est le joueur 1 ou le joueur 2 qui a gagné.
* Avec les if bloquer, on vérifie quel joueur a bloqué l’autre joueur.
* Le « break » sert à quitter la boucle si on a atteint l’objectif.
* Si aucune victoire n'est détectée, c'est le tour du joueur actif.
* On affiche l'état actuel du plateau.
* Ensuite on demande au joueur de choisir un pion à déplacer.
* On lui demande aussi de choisir une direction pour le mouvement.
* On effectue le mouvement du pion choisi par le joueur.
* On change ensuite de joueur pour le prochain tour.
* Enfin on demande à l’utilisateur la taille du plateau de jeu.
* Pour finir, on lance le jeu avec la taille spécifiée.

## La fonction newBoard(n)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Cette fonction crée le plateau de taille n x n.
* Les pions du joueur 2 sont placés dans la première colonne au vertical.
* Les pions du joueur 1 sont placés dans la dernière ligne à l’horizontal.

## La fonction displayBoard(board, n)

Une image contenant texte, capture d’écran

Description générée automatiquement

* Cette fonction affiche le plateau du jeu.
* Les pions du joueur 1 sont les ‘o’ et ceux du joueur 2 sont les ‘x’.
* On crée une liste vide pour stocker les éléments de la ligne actuelle.
* On fait une boucle pour parcourir chaque colonne de la ligne i.
* On vérifie si l'élément à la position (i, j) est égal à 1 (pion du joueur 1).
* On fait pareil pour le joueur 2.
* Si l'élément n'est ni 1 ni 2, cela signifie qu'il est vide.
* On ajoute "." à la liste ligne pour représenter une case vide.
* On affiche la ligne actuelle avec son index (i + 1) et les éléments de la ligne joints par des espaces pour que ce soit plus lisible.
* On affiche une ligne de séparation sous le plateau, avec un trait bas pour chaque colonne.
* Enfin on affiche les numéros de colonne (1 à n) sous le plateau, séparés par des espaces.

## La fonction saisie\_taille\_plateau()

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Cette fonction demande à l’utilisateur de saisir la taille n du plateau pour jouer.
* Tant que la taille du plateau n’est pas un entier et que la taille est inférieure à 2, on redemande à l’utilisateur de saisir à nouveau une taille de plateau.
* On retourne int(taille) qui convertit taille (qui peut être une chaîne de caractères représentant un nombre, par exemple "9") en un nombre entier. Par exemple, si value vaut "9", alors int(value) devient 9.

## La fonction saisie\_position(utilisateur, limite, type\_position)

Une image contenant capture d’écran, texte

Description générée automatiquement

* Cette fonction demande à l'utilisateur de saisir une position, soit ligne soit colonne.
* Tant que la saisie de la valeur n’est pas un chiffre ou alors qu’elle n’est pas entre 1 et la limite qui correspond au nombre n que l’on aura saisie pour la taille du plateau, on redemande à l’utilisateur de saisir à nouveau cette valeur pour la ligne ou la colonne.
* int(value) convertit value (qui peut être une chaîne de caractères représentant un nombre, par exemple "5") en un nombre entier. Par exemple, si value vaut "5", alors int(value) devient 5. Et on lui soustrait 1 car si int(value) vaut 5, l'expression complète int(value) - 1 donnera 4.

## La fonction possiblePawn(board, n, directions, player, i, j)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Cette fonction vérifie si le joueur 1 peut sortir par le haut et si le joueur 2 peut sortir par la droite.
* On vérifie si les coordonnées sont valides et qu'il s'agit d'un pion du joueur.
* On définit le cas où le joueur 1 peut sortir directement par le haut.
* On fait pareil pour le joueur 2 mais cette fois on le fait pour qu’il sorte directement sur la droite.
* On parcourt chaque direction possible.
* On va ensuite calculer la nouvelle position.
* On vérifie si la case ciblée est vide dans le plateau.
* Si c’est vrai donc True le déplacement est possible.
* A l’inverse si c’est faux donc False le déplacement est impossible.

## La fonction selectPawn(board, n, directions, player)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Cette fonction demande à l’utilisateur de sélectionner un pion à déplacer.
* On demande à l’utilisateur de saisir la ligne et la colonne où il veut placer son pion.
* On retourne la position du pion sélectionné par le joueur.
* Si ce n’est pas possible on affiche un message d’erreur au joueur pour qu’il choisisse un autre pion à déplacer.

## La fonction possibleMove(board, n, directions, player, i, j, m)

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

* Cette fonction vérifie si un mouvement est possible pour le pion du joueur à (i, j) dans la direction m.
* Ensuite on obtient le déplacement pour la direction m.
* On va donc calculer la nouvelle position du pion après le déplacement.
* On va ensuite vérifier les conditions de victoire pour le joueur 1 et le joueur 2.
* Donc le joueur 1 peut gagner la partie en sortant du plateau par le haut, alors que le joueur 2 peut gagner en sortant du plateau par le bas.
* Enfin on retourne True si la case ciblée pour sortir est vide.

## La fonction selectMove(board,n,directions,player,i,j)

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

* Cette fonction permet au joueur de choisir une direction de déplacement.
* On initialise 2 listes, une pour les directions empruntées par le joueur 1 et une autre pour les directions empruntées par le joueur 2.
* On fait ensuite une boucle pour demander la direction tant qu’un choix valide n’a pas été fait.
* On demande ensuite au joueur de choisir une direction parmi celles disponibles en fonction de si c’est le joueur 1 ou le joueur 2 qui joue.
* On vérifie ensuite si la direction saisie fait partie des choix possibles pour le joueur.
* Avec la méthode index, on trouve l’indice de la direction choisie.
* On regarde ensuite si le déplacement dans cette direction choisie par le joueur est possible.
* On retourne l'indice de la direction choisie si le déplacement est valide.
* Enfin on renvoie un message d’erreur si le déplacement dans cette direction n’est pas possible.
* Ou alors on informe le joueur que la direction choisie est invalide et on lui demande de réessayer.

## La fonction move(board, n, directions, player, i, j, m)

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

* Cette fonction déplace le pion du joueur de la position (i, j) dans la direction m.
* On enlève le pion du joueur 1 du plateau s'il sort par le haut.
* On enlève le pion du joueur 2 du plateau s'il sort par le bas.
* Ensuite on vérifie que la nouvelle position est valide.
* On enlève le pion de sa position initiale.
* Enfin on place le pion du joueur à la nouvelle position s’il est dans les limites du plateau.

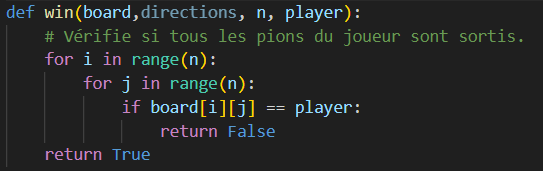
## La fonction bloquer(board, n, directions, player)

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Cette fonction vérifie si le joueur est bloqué (s'il n'a aucun mouvement possible).
* On commence par parcourir chaque ligne et colonne du plateau.
* On vérifie si la case (i, j) contient un pion du joueur.
* Ensuite on appelle la fonction possiblePawn pour vérifier si le pion peut se déplacer.
* On retourne False si le joueur peut déplacer au moins un pion.
* On retourne True si aucun pion du joueur ne peut se déplacer (le joueur est bloqué).

## La fonction win(board, n, directions, player)



* Cette fonction vérifie si le joueur a gagné.
* On regarde si le joueur est le joueur 1.
* On parcourt chaque ligne pour vérifier la première colonne.
* On vérifie si un pion du joueur 1 est encore dans la première colonne.
* On retourne False si le joueur 1 a encore un pion dans la première colonne.
* On retourne True si le joueur 1 n'a aucun pion dans la première colonne (victoire).
* Enfin on fait la même chose pour le joueur 2.