DÉPARTEMENT Sciences Informatiques

DU Développeur niveau II

Algorithmique et Python
Mini projet : puissance 4

Maxence BEKIER

grille_init() : fontion qui renvoie un tableau de 6 lignes et 7 colonnes remplies de zéros

```
fonction grille_init() -> Tableau
variables
    interne
        entier rowIndex # index de déplacement sur les lignes
        entier colIndex # index de déplacement sur les colonnes
    sortie
        tableau grille # Le tableau demandé
début
    grille <- tableau de 6 par 7 # on crée un tableau de 6x7
    pour rowIndex allant de 0 à 6:
        pour colIndex allant de 0 à 7:
            # on prend deux index changeant de valeur entre 0 et les valeurs des lignes et colonnes pour pouvoir se déplacer dans tout le tableau
            grille[colIndex][rowIndex] <- 0 # on donne donc à chaque case la valeur 0
        fin_pour
    fin_pour
    return grille
fin
```

affiche_grille(tableau) : fonction qui affiche la grille du jeu dans la console de la facon la plus esthétique possible

```
fonction affiche_grille(tableau)
variables
    entrée
        tableau tableau # le tableau à afficher
    interne
        entier rowIndex # index de déplacement sur les lignes
        entier colIndex # index de déplacement sur les colonnes
        chaine ligne
début
    afficher("\n| 1 | 2 | 3 | 4 | 5 | 6 | 7 |")
    afficher("-----------------------------")
    pour rowIndex allant de 5 a -1 avec un pas de -1: # on se déplace sur toutes les lignes
        ligne <- "| "
        pour colIndex allant de 0 a 7: # on se déplace sur toutes les colonnes
            ligne <- ligne + chaine(tableau[colIndex][rowIndex]) + " | "
        fin_pour
        afficher("{tableau[rowIndex][colIndex]}") # on affiche donc chaque valeur
    fin_pour
fin
```

colonne_libre(tableau, colonne) : fonction qui renvoie un booléen indiquant s'il est possible de mettre un jeton dans la colonne

```
fonction colonne_libre(tableau, colonne) -> Booléen
variables
    entrée
        tableau tableau # le tableau à vérifier
        entier colonne # la colonne à vérifier
    interne
        entier rowIndex # index de déplacement sur les lignes
début
    pour rowIndex allant de 0 à 6: # on se déplace sur les lignes
        si tableau[colonne][rowIndex] == 0:
            return true # si on rencontre un 0, alors libre
        fin_si
    fin_pour
    return false # si on rencontre pas de 0, alors pas libre
fin
```

place_jeton(tableau, colonne, joueur) : fonction qui place un jeton du joueur (1 ou 2) dans la colonne. Elle renvoie la grille modifiée

```
fonction place_jeton(tableau, colonne, joueur) -> Tableau
variables
    entrée
        tableau tableau # le tableau à modifier
        entier colonne # la colonne à modifier
        entier joueur # le joueur qui place le jeton
    interne
        entier rowIndex # index de déplacement sur les lignes
début
    pour rowIndex allant de 0 a 7: # on se déplace sur les lignes
        si tableau[colonne][rowIndex] == 0:
        # dès qu'on trouve un emplacement libre, on met le pion
            tableau[colonne][rowIndex] <- joueur
            casser_pour
        fin_si
    fin_pour
    return tableau
fin
```

horizontale(tab, joueur) : fonction qui renvoie True si le joueur a au moins 4 jetons alignés dans une ligne

```
fonction horizontale(tableau, joueur) -> Booléen
variables
    entrée
        tableau tableau # le tableau à vérifier
        entier joueur # le joueur dont on vérifie la victoire
    interne
        tableau lignes # tableau représentant la ligne actuelle
        tableau colonnes # tableau représentant la colonne actuelle
        entier colIndex # index de déplacement sur les colonnes
        entier element # index de déplacement sur la colonne
        entier elementIndex # index de déplacement sur element
        entier rowIndex # index de déplacement sur les lignes
début
    # on fait 2 tableau pour prendre les lignes et colonnes
    lignes <- tableau à 1 dimension, de taille 6
    colonnes <- tableau à 1 dimension, de taille 7
    pour colIndex allant de 0 à 7: # on se déplace sur la colonne
        # on ajoute dans le même tableau d'une dimension toutes les
          colonnes
        colonnes[colIndex] <- colonnes[colIndex] + string(tableau[
          colIndex])
    fin_pour
    pour chaque element dans colonnes:
        pour elementIndex allant de 0 à 6:
            # on ajoute dans le même tableau d'une dimension toutes
              les lignes
            lignes[elementIndex] <- lignes[elementIndex] + element[
              elementIndex]
        fin_pour
    fin_pour
    # on cherche si les tableaux contiennent les patternes "1111" ou
      "2222" pour indiquer la victoire des joueurs
    pour chaque rowIndex dans lignes:
        si rowIndex contient "1111" et le joueur == 1:
            return True
        fin_si
        si rowIndex contient "2222" et le joueur == 2:
```

```
            return True
        fin_si
    fin_pour
    return False
fin
```

verticale(tableau, joueur) : fonction qui renvoie True si le joueur a au moins 4 jetons alignés dans une colonne

```
fonction verticale(tableau, joueur) -> Booléen
variables
    entrée
        tableau tableau # le tableau à vérifier
        entier joueur # le joueur dont on vérifie la victoire
    interne
        tableau colonnes # tableau représentant la colonne actuelle
        entier colIndex # index de déplacement sur les colonnes
        entier item # index de déplacement sur la colonne
début
    # on fait 1 tableau pour prendre les colonnes
    colonnes <- tableau à 1 dimension, de taille 7
    pour colIndex allant de 0 à 7: # on se déplace sur la colonne
        # on ajoute dans le même tableau d'une dimension toutes les colonnes
        colonnes[colIndex] <- colonnes[colIndex] + string(tableau[colIndex])
    fin_pour
    # on cherche si les tableaux contiennent les patternes "1111" ou "2222" pour indiquer la victoire des joueurs
    pour chaque item dans colonnes:
        si item contient "1111" et le joueur == 1:
            return True
        fin_si
        si item contient "2222" et le joueur == 2:
            return True
        fin_si
    fin_pour
    return False
fin
```

diagonale(tableau, joueur) : fonction qui renvoie True si le joueur a au moins 4 jetons alignés dans une diagonale

```
fonction diagonale(tableau, joueur) -> Booléen
variables
    entrée
        tableau tableau # le tableau à vérifier
        entier joueur # le joueur dont on vérifie la victoire
    interne
        entier y # index de déplacement sur les lignes
        entier x # index de déplacement sur les colonnes
        entier xx # index de déplacement
        chaine diag # chaine pour détecter les patternes
début
    pour y allant de 0 à 3:
        pour x allant de 0 à 4:
            # on se déplace sur la première partie du tableau avec la courte diagonale
            diag <- ""
            pour xx allant de 0 à 4:
                # on se déplace sur le côté (la diagonale quoi)
                diag <- diag + string(tableau[x+xx][y+xx])
            fin_pour
            # on regarde si le patterne est présent et bon
            si diag == "1111" et joueur == 1:
                return True
            fin_si
            si diag == "2222" et joueur == 2:
                return True
            fin_si
        fin_pour
    fin_pour
    pour y allant de 0 à 3:
        pour x allant de 3 à 7:
            # on se déplace sur la deuxième partie du tableau avec la longue diagonale
            diag <- ""
            pour xx allant de 0 à 4:
                # on se déplace sur le côté (la diagonale quoi)
                diag <- diag + string(tableau[x-xx][y+xx])
```

```
            fin_pour
            # on regarde si le patterne est présent et bon
            si diag == "1111" et joueur == 1:
                return True
            fin_si
            si diag == "2222" et joueur == 2:
                return True
            fin_si
        fin_pour
    fin_pour
fin
```

gagne(tableau, joueur) : fonction qui renvoie True si le joueur a gagné

```
fonction gagne(tableau, joueur) -> Booléen
variables
    entrée
        tableau tableau # le tableau à vérifier
        entier joueur # le joueur dont on vérifie la victoire
début
    # on vérifie si le joueur a 4 pions alignés verticalement, horizontalement ou diagonalement
    return horizontale(tableau, joueur) ou verticale(tableau, joueur) ou diagonale(tableau, joueur)
fin
```

egalite(tableau) : fonction qui renvoie True s'il y a égalité et False sinon

```
fonction egalite(tableau) -> Booléen
variables
    entrée
        tableau tableau # le tableau à vérifier
début
    # on vérifie que la grille soit pleine et qu'aucun joueur n'aie gagné
    return grille_pleine(tableau) et non(gagne(tableau, 1)) et non(gagne(tableau, 2))
fin


fonction grille_pleine(tableau) -> Booléen
variables
    entrée
        tableau tableau # le tableau à vérifier
    interne
        entier rowIndex # index de déplacement de ligne
        entier colIndex # index de déplacement de colonne
début
    pour rowIndex allant de 0 à 6:
        pour colIndex allant de 0 à 7:
            # on se déplace dans tout le tableau avec les index
            si tableau[colIndex][rowIndex] == 0:
                # il ne suffit que d'1 "0" pour dire que la grille n'est pas pleine
                return False
            fin_si
        fin_pour
    fin_pour
fin
```

tour_joueur(tableau, joueur) : fonction qui permet au joueur de placer un jeton dans la colonne choisie. Elle indique si la colonne est pleine et permet alors au joueur de choisir une autre colonne

```
fonction tour_joueur(tableau, joueur) -> Tableau
variables
    entrée
        tableau tableau # le tableau de jeu
        entier joueur # le joueur qui joue
    interne
        entier colonne # colonne où placer son pion
début
    afficher("\nC'est à ton tour joueur {joueur}, dans quelle colonne veux-tu déposer ton pion ?")
    # boucle infinie qui ne sera cassée que quand on le voudra
    tant que 1:
        afficher(">>> ")
        # on demande au joueur la colonne où il veut poser son pion
        colonne <- saisir(int())
        # on vérifie que la colonne est libre
        si colonne_libre(tableau, colonne - 1):
            return place_jeton(tableau, colonne - 1, joueur)
            sortir tant_que # on peut casser la boucle
        sinon:
            # sinon, on lui dit que la colonne est pas libre, et la boucle recommence
            afficher("\nLa colonne {colonne} n'est pas libre ! Choisis-en une autre...")
        fin_si
    fin_tan_que
fin
```

jouer(tableau) : fonction qui permet aux deux joueurs de jouer chacun leur tour. Elle vérifie que les joueurs n'ont pas gagné à la fin de leur tour. Si l'un des deux a gagné ou s'il y a égalité, elle donne le résultat.

```
fonction jouer(tableau)
variables
    entrée
        tableau tableau # le tableau de jeu
début
    # boucle infinie qui ne sera cassée que quand on le voudra
    tant que 1:
        affiche_grille(tableau)
        # c'est le tour du joueur 1
        tableau <- tour_joueur(tableau, 1)
        affiche_grille(tableau)
        # on check pour l'égalité
        si egalite(tableau):
            afficher("Il y a égalité !")
            sortir tant_que # égalité, on casse la boucle
        fin_si
        # on check pour la victoire du joueur 1
        si gagne(tableau, 1):
            afficher("Bravo joueur 1, tu as gagné !")
            sortir tant_que # victoire du j1, on casse la boucle
        fin_si
        # c'est le tour du joueur 2
        tableau <- tour_joueur(tableau, 2)
        affiche_grille(tableau)
        # on check pour l'égalité
        si egalite(tableau):
            afficher("Il y a égalité !")
            sortir tant_que # égalité, on casse la boucle
        fin_si
        # on check pour la victoire du joueur 2
        si gagne(tableau, 2):
            afficher("Bravo joueur 2, tu as gagné !")
            sortir tant_que # victoire du j2, on casse la boucle
        fin_si
    fin_tan_que
fin
```

Programme principal

```
programme principal
variables
    interne
        tableau tableau # le tableau de jeu
début
    # on initialise le tableau de jeu
    tableau <- grille_init()
    # on lance la boucle de jeu
    jouer(tableau)
fin
```