Justine ******, 1D2

BUT Informatique, semestre 1
IUT de Lannion

Année 2022-2023

~ S.A.É. 1.01 - Implémentation d'un besoin client ~

PUISSANCE 4

{Maquette}

{Décomposition algorithmique}

{Cahier de tests}

{Code source intermédiaire}

{Code source final}

{Documentation}

Algorithme du Programme principal

```
programme puissance4 c'est

/* Déclaration des constantes */
     constante entier NB_LIG := 6;
     constante entier NB_COL:= 7;
     constante entier LONGUEUR_MAX_NOM := 30;
     constante entier LONGUEUR_MAX_COULEUR := 10;
     constante entier ALIGNEMENT_GAGNANT := 1;
     constante entier GRILLE_PLEINE := 42;

/* Procédures et fonctions */
     procédure clrscr();
     fonction rand();
     procédure msgChoixCol();
     procédure msgErrSaisie();
     procédure msgErrNumCol();
     procédure msgErrColPleine(entF num_colonne_pleine : entier);
     procédure msgVictoire(entF vainqueur : chaîne, entF loser : chaîne);
     procédure msgMatchNul();
     procédure nomsJoueurs(ent/sortF j_jaune : chaîne, ent/sortF j_rouge : chaîne);
     procédure choixTirage(entF j_jaune : chaîne, entF j_rouge : chaîne, ent/sortF j1 : chaîne,
ent/sortF j1 : chaîne, ent/sortF couleur_j1 : chaîne, ent/sortF couleur_j2 : chaîne);
    procédure msgTour(entF j1 : chaîne, entF j2 : chaîne, entF couleur_j1 : chaîne, entF
couleur_j2 : chaîne, ent/sortF compteur_tour : entier);
    procédure afficheGrilleVide();
    procédure afficheGrilleActu(ent/sortF grille : tableau de chaînes);
    fonction choixColonne(délivre sortF colonne_choisie : entier);
    procédure verifPlaceCol(ent/sortF num_colonne, sortF num_ligne);
    fonction verifAlignement(ent/sortF grille : tableau de chaînes, ent/sortF alignement, entF
j1, entF j2, sortF vainqueur, sortF perdant);

début
/* Déclaration des variables */
     grille : tableau[0..NB_LIG, 0..NB_COL] de chaînes;          // grille du jeu

     jaune : chaîne(LONGUEUR_MAX_NOM);                          // prénom du joueur jaune
     rouge : chaîne(LONGUEUR_MAX_NOM);                          // prénom du joueur rouge
     joueur1 : chaîne(LONGUEUR_MAX_NOM);                        // prénom du joueur 1
     joueur2 : chaîne(LONGUEUR_MAX_NOM);                        // prénom du joueur 2
     couleur_j1 : chaîne(LONGUEUR_MAX_COULEUR);                 // couleur du joueur 1
     couleur_j2 : chaîne(LONGUEUR_MAX_COULEUR);                 // couleur du joueur 2
     justine : chaîne(LONGUEUR_MAX_NOM);                        // prénom du vainqueur
     arthur : chaîne(LONGUEUR_MAX_NOM);                         // prénom du loser

     compteur_tour : entier;              // compte le nombre de tours
     num_colonne : entier;                // numéro de la colonne choisie
     alignement : entier;                 // met fin à la partie si égal à ALIGNEMENT_GAGNANT

/* Initialisation des variables */
     compteur_tour := 0;
     alignement := 0;
```

```
/* Corps du programme */

    nomsJoueurs(ent/sortE jaune, ent/sortE rouge);
    choixTirage(entE jaune, entE rouge, ent/sortE joueur1, ent/sortE joueur2, ent/sortE
couleur_j1, ent/sortE couleur_j2);

     tantque ((alignement == 0) ET (compteur_tour < GRILLE_PLEINE)) faire

            msgTour(entE j1, entE j2, entE couleur_j1, entE couleur_j2, ent/sortE
compteur_tour);
            si (compteur_tour == 1) alors
                    afficheGrilleVide();
            sinon
                    verifPlaceCol(ent/sortE num_colonne, sortE num_ligne);
                    verifAlignement(ent/sortE grille, ent/sortE alignement, entE j1, entE j2,
sortE vainqueur, sortE perdant);
                    afficheGrilleActu(ent/sortE grille);
            finsi
            num_colonne := choixColonne();
    finfaire

    si (alignement == ALIGNEMENT_GAGNANT) alors
            msgVictoire();
    finsi

    si ((alignement == 0) ET (compteur_tour == GRILLE_PLEINE)) alors
            msgMatchNul();
    finsi
fin
```

Liste des Procédures & Fonctions

Type	Nom	Paramètres	Résultat	Description
P	**afficheGrilleActu**	variable : **grille** mode : entrée/sortie type : tableau de chaînes rôle : représente la grille de la partie en cours		***Affiche la grille du jeu actualisée.***
P	**afficheGrilleVide**			***Affiche la grille du jeu vide.***
F	**choixCol**		variable : **num_colonne** mode : sortie type : entier rôle : numéro de la colonne choisi par le joueur durant son tour	***Enregistre la saisie du joueur dans la variable num_colonne.*** ⇒ *La fonction ne prend aucun paramètre en entrée, mais renvoie la valeur de num_colonne. En cas de mauvaise saisie, elle appelle la procédure msgErrNumCol() pout afficher un message d'erreur et redemande à l'utilisateur de choisir une colonne.* -Affiche : "👉 " pour inviter le joueur à saisir le numéro d'une colonne

Type	Nom	Paramètres	Résultat	Description
P	**choixTirage**	variable : **jaune** mode : entrée type : chaîne(30) rôle : nom du joueur jaune variable : **rouge** mode : entrée type : chaîne(30) rôle : nom du joueur rouge variable : **joueur1** mode : entrée/sortie type : chaîne(30) rôle : nom de joueur qui commencera la partie variable : **joueur2** mode : entrée/sortie type : chaîne(30) rôle : nom du second joueur variable : **couleur_j1** mode : entrée/sortie type : chaîne(10) rôle : couleur du joueur 1 variable : **couleur_j2** mode : entrée/sortie type : chaîne(10) rôle : couleur du joueur 2		***Détermine quel joueur (nom et couleur) commence la partie, en laissant le choix d'un tirage aléatoire ou non.*** ⇒ *La procédure reçoit le nom des deux joueurs et leur couleur respective en entrée, ainsi que les variables **joueur1**, **joueur2**, **couleur_j1** et **couleur_j2** (en entrée/sortie). En fonction du choix de l'utilisateur (aléatoire ou non), la procédure copie le nom du joueur qui commence dans **joueur1** et sa couleur dans **couleur_j1**. De même pour le joueur 2.* - Affiche le message : "🏁 Qui va commencer ? 🏁" - Si l'utilisateur saisie 1, le joueur jaune commencera - Si l'utilisateur saisie 2, le joueur rouge commencera - Si l'utilisateur saisie 0, une fonction aléatoire tire au sort le joueur qui commence - Si l'utilisateur saisie une valeur non proposé, on appelle la procédure **msgErreurSaisie()** puis demande de saisir un nouveau choix
P	**clrscr**			***Vide la fenêtre du terminal***

Type	Nom	Paramètres	Résultat	Description
P	**msgChoixCol**			*Informe le joueur dont c'est le tour qu'il doit jouer (s'affiche uniquement aux deux premiers tours).* - Affiche : "👇 Choisis la colonne dans laquelle tu souhaites insérer ta pièce 👇" - Affiche : "(Saisis un chiffre entre 1 et 7)"
P	**msgErrColPleine**	variable : **num_colonne_pleine** mode : entrée type : entier rôle : numéro de la colonne pleine choisie		*Informe le joueur que la colonne choisie est déjà pleine* - Affiche le message "Impossible d'ajouter de nouveaux jetons en <numéro_colonne_pleine>, cette colonne est déjà pleine !"
P	**msgErrNumCol**			*Informe le joueur que le numéro de colonne saisi n'est pas valide.* - Affiche le message "Bien essayé, mais cette colonne n'existe pas…! 🤡"
P	**msgErrSaisie**			*Informe le joueur d'une saisie erronée dans le choix de la colonne.* - Affiche le message "😕 Désolé, je n'ai pas compris ta réponse…"
P	**msgMatchNul**			*Notifie les joueurs d'un match nul* - Affiche le message "Match nul ! Félicitations à tous les deux ! 🤝"

Type	Nom	Paramètres	Résultat	Description
P	**msgTour**	variable : **j1** mode : entrée type : chaîne(30) rôle : nom de joueur 1 variable : **j2** mode : entrée type : chaîne(30) rôle : nom du joueur 2 variable : **couleur_j1** mode : entrée type : chaîne(10) rôle : couleur du joueur 1 variable : **couleur_j2** mode : entrée type : chaîne(10) rôle : couleur du joueur 2 variable : **compteur_tour** mode : entrée/sortie type : entier rôle : compte le nombre de tours de la partie		***Informe le joueur dont c'est le tour qu'il doit placer une pièce dans la grille.*** ⇒ *La procédure reçoit les noms et les couleurs des joueurs en paramètres d'entrée (**j1**, **j2**, **couleur_j1** et **couleur_j2**) ainsi que le compteur de tours (**compteur_tour**) en entrée/sortie et affiche un message différent en fonction du tour en cours, déterminé grâce à la variable **compteur_tour** qui s'incrémente de 1 à chaque tour. On appelle la procédure **msgChoixCol()** uniquement au deux premiers tours.* - Au premier tour du joueur 1, affiche le message : "`C'est <j1> qui commence !`", puis appelle **msgChoixCol()** - Au premier tour du joueur 2 (deuxième tour), affiche le message : "`C'est ton tour, <j2> !`", puis appelle **msgChoixCol()** - Aux tours suivants, affiche le message : "`À toi, <joueur_dont_c'est_le_tour> !` 👇"
P	**msgVictoire**	variable : **vainqueur** mode : entrée type : entier rôle : nom du gagnant de la partie variable : **loser** mode : entrée type : entier rôle : nom du perdant de la partie		***Notifie le joueur qui a gagné*** - Affiche : "`<vainqueur> remporte la partie !` 🥳" - Affiche : "`<loser>, sans rancune ! Puis, la deuxième place, c'est pas si mal…`"

Type	Nom	Paramètres	Résultat	Description
P	**nomsJoueurs**	variable : **jaune** mode : entrée/sortie type : chaîne(30) rôle : nom du joueur jaune variable : **rouge** mode : entrée/sortie type : chaîne (30) rôle : nom du joueur rouge		***Demande aux deux joueurs qui vont s'affronter de saisir leurs noms et de choisir la couleur des jetons qu'ils vont jouer.*** ⇒ *La procédure reçoit en paramètres d'entrée/sortie les deux variables des couleurs (**jaune** et **rouge**) et attribue à chacune le nom d'un joueur.* - Appel de la fonction **clrscr** pour vider la fenêtre - Affiche le message : "⚔️ Qui va s'affronter ? ⚔️" - Affiche le message : "🟡 Joueur jaune, écris ton prénom : " - Enregistre le nom du premier joueur dans la variable **jaune** - Affiche le message : "🔴 Joueur rouge, écris ton prénom : " - Enregistre le nom du second joueur dans la variable **rouge**
F	**rand**		variable : **tirage** mode : sortie type : entier rôle : nombre généré aléatoirement utilisé dans le cas où les joueurs choisissent un tirage au sort	***Fonction qui renvoie un entier aléatoire.***

Type	Nom	Paramètres	Résultat	Description
P	**verifAlignement**	variable : **grille** mode : entrée type : tableau de chaînes rôle : représente la grille de la partie en cours variable : **j1** mode : entrée type : chaîne(30) rôle : nom de joueur 1 variable : **j2** mode : entrée type : chaîne(30) rôle : nom du joueur 2 variable : **justine** mode : sortie type : chaîne(30) rôle : nom du gagnant variable : **arthur** mode : sortie type : chaîne (30) rôle : nom du perdant		**Vérifie qu'aucun alignement gagnant n'a été fait.** ⇒ Cette procédure parcours la grille dans tous les sens d'alignement possible : - horizontalement : vérifie si 4 cellules d'une même colonne sont occupées successivement par des jetons d'une même couleur. - verticalement : vérifie si 4 cellules d'une même ligne sont occupées successivement par des jetons d'une même couleur. - en diagonal : vérifie pour chaque cellule de coordonnées (x = [5 à 3] ; y = [0 à 3]) (avec x ⇔ **ligne** et y ⇔ **colonne**) si les trois cellules suivantes (x-1 ; y+1), (x-2 ; y+2), (x-3 ; y+3) sont occupées par des jetons d'une même couleur. De même pour (x = [5 à 3] ; y = [3 à 6] et leurs trois cellules précédentes (x-1 ; y-1), (x-2 ; y-2), (x-3 ; y-3) Dès qu'un alignement est trouvé, alignement prend la valeur de 1 et on sort de la boucle. **justine** prend la valeur du nom du joueur dont les pièces ont été alignées, **arthur** prend celle du nom du perdant.
P	**verifPlaceCol**	variable : **num_colonne** mode : entrée type : entier rôle : colonne choisie par le joueur variable : **grille** mode : entrée/sortie type : tableau de chaînes rôle : représente la grille de la partie en cours		***Vérifie pour un numéro de colonne donné, num_colonne, que celle-ci n'est pas pleine et détermine la cellule à laquelle placer la pièce du joueur.*** ⇒ *La procédure reçoit le tableau de la grille en entrée/sortie et le numéro de colonne en entrée. Pour chaque ligne en partant du bas-gauche de la grille, elle vérifie si la celulle de coordonnées (**ligne** = **NB_LIGNES** - 1 ; **colonne** = **num_colonne**) contient un jeton. Tant qu'une cellule vide n'a pas été trouvée, elle continue en décrémentant la ligne de 1 à chaque tour. Dès qu'une cellule vide est trouvée, on place le jeton du joueur dont c'est le tour et on sort de la boucle. Si aucune cellule vide n'est trouvée, cela signifie que la colonne est pleine. On appelle alors la procédure **msgErrColPleine()** et on invite le joueur à saisir le numéro d'une nouvelle colonne.*