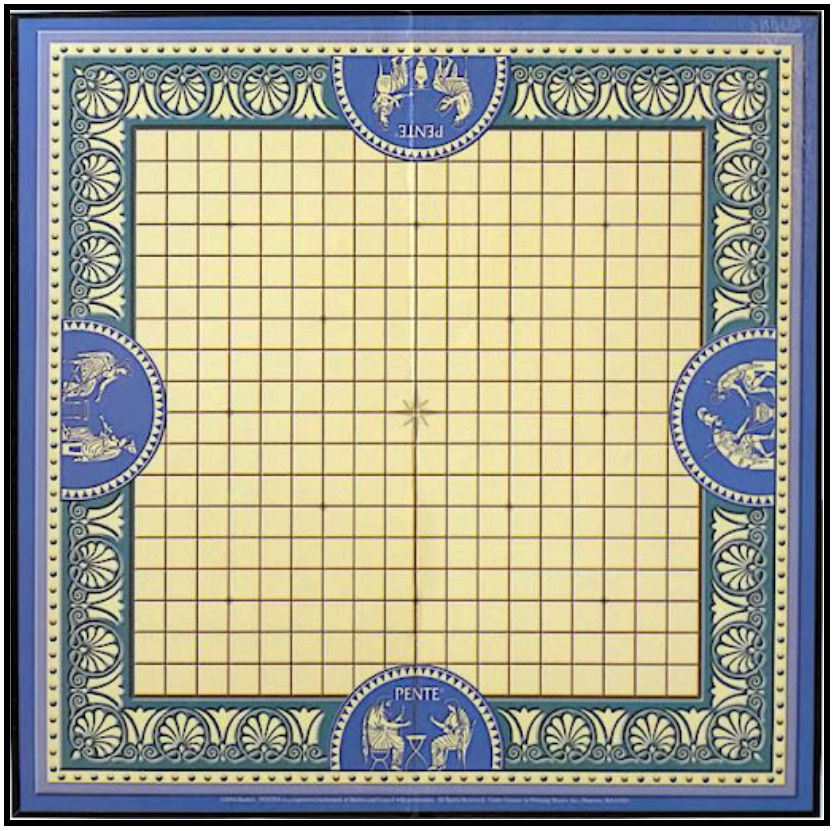
POLYTECH LILLE

Université de Lille

Algorithmique et Programmation

Rapport du projet



Tuteur: LIETARD Thibault

HANI Nabil

TASSA Hlba

23/01/2025

Sommaire

1- Introduction

Dans le cadre d'Algorithmique et Programmation, il nous a été proposé de faire un projet consistant à développer un programme en C pour obtenir un jeu de de société "Penté". Ce jeu stratégique est joué sur un plateau de 19x19 cases avec des règles simples, mais nécessitant une grande intelligence dans les tactiques.

Ce jeu de société est composé de deux joueurs, X et O, qui placent leurs pions sur le plateau à tour de rôle et qui doivent essayer de gagner de plusieurs manières différentes. Le joueur peut gagner par:

- Alignement: Ici, il doit aligner ces cinq pions verticalement, horizontalement, ou diagonalement.

- Captures: Lorsque deux pions du joueur (dont le pion qui vient d'être joué) entourent deux pions de l'adversaire, ces deux pions adverses vont disparaître du plateau et deux points vont être attribués au joueur. Il est important de noter que les prises peuvent être effectuées diagonalement aussi. Le joueur peut aussi capturer plusieurs paires de pions adverses en même temps. Si un joueur réussit à capturer dix pions adverses, il est automatiquement déclaré comme gagnant

- Abandon: Un joueur peut simplement abandonner au milieu de la partie, ce qui veut dire que le joueur adversaire sera déclaré avec la victoire.

L'objectif principal de ce projet est de rédiger un programme en C qui respecte les règles fondamentales de ce jeu expliquées ci-dessus, tout en garantissant aux joueurs (dans notre cas, l'utilisateur) une expérience fluide et conforme au cahier des charges. Concernant le rapport, il est demandé de détailler les différentes étapes du développement du programme avec des réflexions critiques sur nos choix effectuées lors de ce projet. On doit aussi parler des défis qu'on a rencontrés durant la conception de ce projet.

2- Cahier des charges

2.1- Modélisation du jeu

Le plateau de jeu est représenté par un tableau de caractères 19×19, où chaque case peut contenir un point "." (ce qui veut dire que la case est vide), X (croix) ou O (rond). Ce tableau constitue l'élément central de la modélisation du jeu.

La structure Jeu permet de suivre l'état de la partie. Elle inclut le plateau, deux scores (scoreX et scoreO) initialisés à zéro pour compter les pions capturés, et une variable tour qui gère le joueur actif (-1 pour O et 1 pour X).

Les principales règles du jeu sont implémentées à travers des fonctions : verif_captures pour détecter et gérer les prises, verif_alignement pour vérifier les alignements de cinq pions ou plus, et tour_joueur pour alterner les joueurs tout en contrôlant les conditions de victoire.

Enfin, le déroulement du jeu est accompagné d'un affichage dynamique via la fonction affichage_plateau, qui montre clairement l'état actuel du jeu et les scores des deux joueurs.

2.2- Analyse

Le programme doit gérer plusieurs aspects clés pour respecter ces règles. Il doit permettre aux joueurs de saisir leurs coups tout en vérifiant leur validité (case libre et dans les limites du plateau). Après chaque coup, il faut vérifier si une capture ou un alignement a eu lieu, et mettre à jour l'état du jeu en conséquence. Enfin, le plateau doit être affiché de manière claire à chaque tour, avec les scores de chaque joueur et l'état des cases. Cette analyse permet de bien cerner les exigences et de structurer les étapes pour concevoir une solution robuste.

2.3- Contraintes et limites

Dans ce projet, on a plusieurs contraintes à respecter dans différents aspects. Voici ces contraintes structurés pour faciliter la compréhension de l'algorithme, résumant ce qui a été dit précédemment:

- **Affichage du plateau:** Lors du débogage du programme, on doit afficher correctement le plateau après chaque coup en respectant le format spécifié dans l'énoncé (des '.' sur les cases vides et des symboles 'X' ou 'O' pour les joueurs, avec le numéro de ligne et de colonne aux bords). Il faut aussi afficher le nombre de pions capturés par chaque joueur.
- **Saisie des coups:** L'utilisateur doit saisir les coups sous le format 'ligne,colonne' avec une validation de cette saisie pour éviter les placement de pions dans des cases occupées ou hors plateau.
- **Conditions de victoire:** Il est essentiel d'implémenter correctement les conditions de victoire (alignement et captures)

- **Abandon:** Le programme doit directement détecter l'abandon d'un joueur lorsqu'il saisit les indices 0,0.
- **Fichiers de tests:** Le programme doit fonctionner correctement avec les fichiers tests (pente1.txt et pente2.txt) et doit correspondre exactement aux scénarios attendus (victoire par alignement pour pente1.txt et victoire par abandon de l'adversaire pour pente2.txt)

Avec ces contraintes respectées, on peut avoir quelques limites qui méritent d'être analysées afin d'assurer une conformité complète, ainsi qu'une expérience optimale de la part de l'utilisateur. Dans ces limites, on retrouve:

- **Ergonomie:** En termes d'ergonomie, bien que l'interface soit fonctionnelle, elle pourrait être améliorée esthétiquement. Par exemple, on peut améliorer l'interface en ayant une représentation plus concrète et visuelle du plateau du jeu de société "Penté". De plus, il serait mieux d'afficher le plateau une seule fois au début de la partie et de le mettre à jour dynamiquement à chaque tour, plutôt que de le réafficher après chaque coup. On peut aussi ajouter un guide complet avant le début de la partie pour mieux orienter les joueurs en leur fournissant des instructions claires sur les règles du jeu.
- **Gestion des exceptions:** Le programme ne contient pas de mécanisme qui permet de continuer le jeu s'il y a eu des erreurs inattendues. C'est simplement un jeu sur une interface de terminal qui peut s'arrêter facilement de plusieurs façons.
- **Tests étendus:** Le programme doit être testé pour couvrir des scénarios maximaux. Donc, on va rajouter des fichiers tests qui permettent de tester les scénarios suivants: Victoire par alignement horizontal, vertical, diagonal montant, diagonal descendant et victoire par captures.

3- Analyse du problème

3.1- Contexte

Maintenant, dans un aspect de programmation, on doit développer un programme en langage C qui nous permettra de jouer sur un plateau de 19x19 cases Les objectifs principaux sont le respect de ces 3 règles suivantes: Gagner par alignement, gagner par captures et gagner par abandon. On se concentre sur ces 3 puisque ce sont les seuls moyens qu'une partie puisse se terminer. Mais il faut aussi assurer que le jeu soit interactif et que les deux joueurs puissent jouer de manière équilibrée. Mais dans notre cas, le premier joueur à déposer son pion ne sera pas décidé de manière équilibrée, c'est le joueur avec le symbole 'O' qui commence.

3.2- Méthode de résolution

Donc, en premier, on doit représenter le plateau avec un tableau 2D (on peut l'appeler matrice) de caractères de dimension 19 x 19 .L a dimension sera déclarée en tant que constante nommée A (on le met en début de programme en écrivant '#define A 19').

Après, on va déclarer une structure qu'on appellera Jeu de la manière suivante:

```
typedef struct {
    char plateau[A][A];
    int scoreX;
    int scoreO;
    int tour;
} Jeu;
```

Cette structure de données contient:

- plateau: matrice de caractères représentant l'état du jeu (mise à jour de la matrice effectuée après chaque saisie de l'utilisateur)
- scoreX: Entier indiquant le score du joueur avec le symbole 'X'.
- scoreO: Entier indiquant le score du joueur avec le symbole 'O'.
- Tour: Entier permettant d'alterner les tours de joueurs. On associera l'entier -1 au joueur avec le symbole 'O' et l'entier 1 au joueur 'X' (puisqu'on commencera à -1 pour O, puis on passera à 1 pour X en multipliant -1 par -1 et on reviendra à O en multipliant 1 par -1, vice versa)

Maintenant, commençons à expliquer les algorithmes principaux qui nous aideront à concevoir notre jeu:

Initialisation du jeu: Quand la partie commence, cet algorithme permettra d'initialiser le plateau avec des cases vides qui seront représentées avec des points. On va également initialiser le score des deux joueurs à 0, et enfin la variable tour sera initialisée à -1 (puisque le joueur 'O' commence).

Affichage du plateau: Après avoir initialisé ces variables, cet algorithme permettra d'afficher le plateau et les scores des deux joueurs (ils seront bien sûr mis à jour après chaque coup).

Validation des saisies de coups: Quand un joueur saisit les indices du pion à déposer, un algorithme sera conçu pour détecter si la case saisie est libre et que la case se situe dans le plateau. Si c'est le cas, le symbole du joueur sera rajouté sur la matrice du plateau, dans la case spécifiée.

Détection d'alignement: Cet algorithme permettra de détecter un alignement dans toutes les directions

Captures: Dans cet algorithme, on pourra identifier les pions adverses qui sont entourés, puis les retirer.

Abandon: Cet algorithme détectera simplement la saisie de l'indice 0,0.

4- Explication détaillée des sous-programmes

4.1- void init_jeu

L'action init_jeu initialise les éléments nécessaires pour lancer la partie du jeu.

```
/*****************************************************************************************************/
/* Init_jeu : Action d'initialisation de la structure jeu : plateau vide, les scores = 0 et le tour au pion rond O */
/* pente  : Jeu - Resultat                                                                           */
/*****************************************************************************************************/

void init_jeu (Jeu *pente)
{
    for (int i = 0; i < A; i++)
    {
        for (int j = 0; j < A; j++)
        {
            pente -> plateau[i][j] = '.';
        }
    }
    pente -> scoreX = 0;
    pente -> scoreO = 0;
    pente -> tour = -1;  //-1 pour O et 1 pour X
}
```

Ici, on a comme paramètre d'entrée un pointeur vers une structure Jeu nommé pente (Jeu *pente). Ce pointeur contient les variables dans la structure montrée précédemment. Ici on ne retourne rien puisque c'est une action.

En premier, on commence par initialiser le plateau avec une boucle qui traverse chaque ligne avec les i et chaque colonne avec les j. À chaque itération, la case correspondante (ici c'est plateau[i] [j]) sera mise à jour avec un '.'.

Ensuite, les variables 'scoreX' et 'scoreO' sont initialisées à 0 puisque ces dernières comptent le nombre de pions capturés respectivement par les joueurs X et O en début de jeu (ce qui explique le nom de l'action).

Enfin, la variable 'tour' est initialisée à -1 puisque le joueur 'O' commence. Cette variable permet d'alterner correctement les tours.

4.2- float verif_alignement

La fonction verif_alignement est utilisée pour vérifier si un joueur a pu aligner cinq pions consécutifs de son symbole dans n'importe quelle direction.

```c
/*****************************************************************************************************/
/* verif_alignement : fonction pour verifier l'alignenment de 5 pions de meme symbole dans tous les directions */
/* pente : Jeu - Le jeu qu'on verifie - Donnee                                                       */
/* symbole : Chaîne de caractères - Le joueur qui vient de placer un pion - Locale                   */
/*****************************************************************************************************/

float verif_alignement (Jeu pente)
{
    char symbole ;
    if (pente.tour == 1)
    {
        symbole = 'X';
    }else
    {
        symbole = 'O';
    }
    for (int i = 0; i < A; i++)
    {
        for (int j = 0; j < A; j++)
        {
            if (pente.plateau[i][j] == symbole)
            {
                // Alignement horizontal
                if (j <= A - 5 && pente.plateau[i][j+1] == symbole &&
                    pente.plateau[i][j+2] == symbole && pente.plateau[i][j+3] == symbole &&
                    pente.plateau[i][j+4] == symbole)
                    {
                        return 1;
                    }

                //Alignement vertical
                if (i <= A-5 && pente.plateau[i+1][j] == symbole && pente.plateau[i+2][j] == symbole &&
                    pente.plateau[i+3][j] == symbole && pente.plateau[i+4][j])
                    {
                        return 1;
                    }

                //Alignement diagonal (descendant)
                if (i <= A-5 && j <= A-5 && pente.plateau[i+1][j+1] == symbole &&
                    pente.plateau[i+2][j+2] == symbole && pente.plateau[i+3][j+3] == symbole &&
                    pente.plateau[i+4][j+4] == symbole)
                    {
                        return 1;
                    }

                //Alignement diagonal (ascendant)
                if (i >= 4 && j <= A-5 && pente.plateau[i-1][j+1] == symbole && pente.plateau[i-2][j+2] == symbole &&
                    pente.plateau[i-3][j+3] == symbole && pente.plateau[i-4][j+4] == symbole)
                    {
                        return 1;
                    }
            }
        }
    }
    return 0;
}
```

Dans cette fonction, on prend comme paramètre d'entrée la structure Jeu et on retourne 1 si un alignement de cinq pions est détecté, 0 sinon.

En premier, le symbole du joueur est déterminé en fonction de la variable tour: Si la variable tour est égale à 1, on prendra le symbole X, sinon, si elle est égale à -1, on prendra le symbole O.

Ensuite, on parcourt le plateau avec deux boucles (une pour les lignes et l'autre pour les colonnes) pour voir si la case courante contient le symbole du joueur actif. Si c'est le cas, on commencera à vérifier s'il y a un alignement dans les quatres directions à partir de ce symbole détecté dans l'indice i,j.

Pour l’alignement vertical, on vérifie s’il y a cinq cases consécutives sur la même ligne qui contiennent le symbole du joueur actif, tout en garantissant qu’on ne dépasse pas les limites du plateau (j <= A - 5)

Quant à l’alignement horizontal, on vérifiera s’il existe cinq cases consécutives dans la même colonne contenant le symbole du joueur actif, avec la condition i <= A - 5 vérifiée (pour rester dans les limites du plateau).

Puis pour l’alignement diagonal descendant, on doit essayer de trouver cinq cases consécutives dans une diagonale descendante contenant le symbole du joueur actif. Dans une diagonale descendante, on monte d’une ligne (i+1) et d’une colonne (j+1) à chaque fois. On doit aussi vérifier les conditions suivantes: i <= A - 5 et j <= A - 5 pour ne pas sortir du plateau.

Enfin, pour l’alignement diagonal ascendant/montant, on va voir si on peut trouver cinq cases consécutives dans une diagonale montante avec le symbole du joueur actif. Pour parcourir les cinq cases dans une diagonale ascendante, on descend d’une ligne (i-1) et on monte d’une colonne (j+1). On doit également vérifier les conditions suivantes:

- i >= 4 puisqu’on part de i-1 jusqu’à i-4
- j <= A - 5 pour vérifier qu’on ne dépasse pas les bords du côté droit.

Si aucune des conditions d'alignement est vérifiée, la fonction retournera 0.

4.3- void verif_captures (avec algorithme)

Cette action permet de vérifier si un coup joué par le joueur actif permet de capturer des pions adverses. Si une capture est effectuée, le plateau et les scores sont directement mis à jour.

```c
void verif_captures(Jeu *pente, int ligne, int colonne)
    char joueur;
    char adversaire;

    if (pente -> tour == 1)
    {
        joueur = 'X';
        adversaire = 'O';
    }else
    {
        joueur = 'O';
        adversaire = 'X';
    }

    int directions[8][2] = {
        {0, 1},   // Droite
        {0, -1},  // Gauche
        {1, 0},   // Bas
        {-1, 0},  // Haut
        {1, 1},   // Diagonale Bas Droite
        {-1, -1}, // Diagonale Haut Gauche
        {1, -1},  // Diagonale Bas Gauche
        {-1, 1}   // Diagonale Haut Droite
    };

    for (int d = 0; d < 8; d++) {
        int dx = directions[d][0], dy = directions[d][1];
        int x1 = ligne + dx, y1 = colonne + dy;
        int x2 = ligne + 2 * dx, y2 = colonne + 2 * dy;
        int x3 = ligne + 3 * dx, y3 = colonne + 3 * dy;
        if (x1 >= 0 && x1 < A && y1 >= 0 && y1 < A &&
            x2 >= 0 && x2 < A && y2 >= 0 && y2 < A &&
            x3 >= 0 && x3 < A && y3 >= 0 && y3 < A &&
            pente->plateau[x1][y1] == adversaire &&
            pente->plateau[x2][y2] == adversaire &&
            pente->plateau[x3][y3] == joueur)
        {
            pente->plateau[x1][y1] = '.';
            pente->plateau[x2][y2] = '.';
            if (joueur == 'X')
            {
                pente->scoreX += 2;
            } else
            {
                pente->scoreO += 2;
            }
        }
    }
}
```

Elle prend en entrée un pointeur vers la structure Jeu, deux entiers qui représentent la ligne et la colonne du pion joué. En sortie, on retourne rien puisque c'est une action

En premier, on identifie le joueur actif et le joueur inactif. Ici on a besoin du joueur inactif puisqu'on doit détecter son symbole aussi. Donc si la variable tour est égale à 1, le joueur actif sera X et le joueur inactif sera O et si c'est égale à -1, on inversera les symboles du joueur actif et du joueur inactif.

La variable 'int directions[8][2]' permet d'analyser toutes les directions possibles de captures (à partir de la position du pion qui vient d'être joué par le joueur actif bien sûr).

Ensuite, on fait une boucle qui parcourt chaque direction possible (avec l'indice d qui va de 0 à 7) à partir du pion joué. On utilise les variables dx et dy puisqu'ils représenteront respectivement le déplacement en ligne et en colonne.

À partir de la position initiale du pions (ici le pion joué sur l'indice ligne,colonne), on va identifier les positions des trois cases consécutives dans la direction d:
(x1,y1) sera la première case, (x2,y2) la deuxième et (x3,y3) la troisième.
Après, on vérifie si toutes les cases mentionnées ci-dessus sont toutes à l'intérieur du plateau. Si elles le sont, on passera à une autre vérification.

On vérifiera ensuite si les cases (x1,y1), (x2,y2) contiennent des pions adverses et si la case (x3,y3) contient un pion du joueur actif. Si toutes ces conditions sont vérifiées, les deux cases

d'indice (x1,y1) et (x2,y2) dans le plateau seront remplacées par un point, puisqu'elles seront capturées. L'ajout de 2 points sera ensuite ajouté sur le score du joueur actif.

Pour mieux expliquer comment ça fonctionne, on va prendre un exemple. Imaginons que le joueur actif est le joueur O et il pose un pion dans la case (9,6). Prenons la direction (1,0) (donc on vérifiera si la capture est possible vers le bas). On a dx = 1 et dy = 0. Donc les cases vérifiées seront:

- (x1,y1) = (10,6)
- (x2,y2) = (11,6)
- (x3,y3) = (12,6)

Si (10,6) et (11,6) contiennent le symbole X et la case (x3,y3) contient le symbole X, une prise des deux symboles X sera effectuée et on ajoutera deux points au score du joueur O.

Voici l'algorithme de l'action verif_captures qui gère la prise des pions:

```
Action verif_captures(pente, ligne, colonne) :
    Donnée/Résultat : pente : Jeu
    Donnée : ligne  : Entier
             colonne : Entier
    Locales : joueur : Caractère
              adversaire : Caractère
              direction : matrice[8][2] d'entiers
              dx , dy : entiers
              x1 , x2 , x3 : entiers
              y1 , y2 , y3: entiers

    {initialisation de la matrice}
    direction [1][1] ← 0
    direction [1][2] ← 1
    direction [2][1] ← 0
    direction [2][2] ← 1
    direction [3][1] ← 1
    direction [3][2] ← 0
    direction [4][1] ← 1
    direction [4][2] ← 0
    direction [5][1] ← 1
    direction [5][2] ← 1
    direction [6][1] ← 1
    direction [6][2] ← 1
```

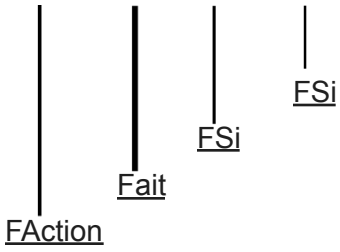
```
direction [7][1] ← 1
direction [7][2] ← 1
direction [8][1] ← 1
direction [8][2] ← 1

{ test du joueur actif }
Si (pente.tour = 1) Alors
        joueur ← X
        adversaire ←  O
Sinon Si ( pente.tour = 0 ) Alors
            joueur <- O
            adversaire <- X
      FSi
FSi

Pour d de 1 à 8 Faire
        dx = direction [d][1]
        dy = direction [d][2]
        x1 = ligne + dx
        y1 = colonne + dy
        x2 = ligne + 2*dx
        y2 = colonne + 2*dy
        x3 = ligne + 3*dx
        y3 = colonne + 3*dy

        Si ((x1 > 0) et (x1 < A) et (y1 >= 0) et (y1 < A) et (x2 >= 0) et (x2 < A) et (y2>= 0)
        et (y2 < A) et (x3 >= 0) et (x3 < A) et (y3 >= 0) et (y3 < A) et
        (pente.plateau[x1][y1] = adversaire) et (pente.plateau[x2][y2] = adversaire) et
        (pente.plateau[x3][y3] = joueur) ) Alors

            pente.plateau[x1][y1] = .
            pente.plateau[x2][y2] = .
            Si (joueur = X) Alors
                pente.scoreX = pente.score + 2
            Sinon   Si (joueur = O) Alors
                        pente.scoreO = pente.scoreO + 2
                    FSi
```


FSi
FSi
Fait
FAction

4.4- void tour_joueur

L'action tour_joueur est le sous-programme le plus important puisque ça gère tout le déroulement du jeu. C'est grâce à cette action qu'on pourra alterner de tours, mettre à jour le plateau, détecter une victoire, etc…

```c
void tour_joueur(Jeu *pente, int *fini)
{
    int ligne;
    int colonne;
    char joueur;
    char adversaire;

    if (pente->tour == 1)
    {
        joueur = 'X';
        adversaire = 'O';
    }
    else
    {
        joueur = 'O';
        adversaire = 'X';
    }

    if (pente->scoreO >= 10)
    {
        affichage_plateau(*pente);
        printf("Joueur O gagne par captures\n");
        *fini = 1;
    }
    else if (pente->scoreX >= 10)
    {
        affichage_plateau(*pente);
        printf("Joueur X gagne par captures\n");
        *fini = 1;
    }

    printf("Joueur de symbole %c: Indiquez l'indice (ligne, colonne) où vous voulez placer votre pion (entrez 0,0 pour abandonner).", joueur);
    scanf("%d,%d", &ligne, &colonne);

    if (ligne == 0 && colonne == 0)
    {
        printf("Le joueur %c a abandonné et le joueur %c a gagné \n", joueur, adversaire);
        *fini = 1;
    }

    if (ligne < 1 || ligne > A || colonne < 1 || colonne > A ||
        pente->plateau[ligne - 1][colonne - 1] != '.')
    {
        if(ligne != 0 || colonne != 0)
        {
            printf("Veuillez réessayer. \n");
        }
    }
    else
    {
        pente->plateau[ligne - 1][colonne - 1] = joueur;
        verif_captures(pente, ligne - 1, colonne - 1);
    }

    if (verif_alignement(*pente))
    {
        affichage_plateau(*pente);
        printf("Le joueur portant le symbole %c a gagné par alignement \n", joueur);
        *fini = 1;
    }

    pente->tour *= -1;
}
```

On prend en entrée un pointeur vers la structure Jeu (permettra de faire les modifications nécessaires dans le plateau ou dans les scores) et un pointeur vers un entier (*fini représente l'état de la partie: si elle est égale à 1, la partie est terminée et si elle est égale à 0, on continue)

En premier, on vérifie qui est le joueur actif et qui est le joueur inactif de la même façon que dans l'action verif_captures. Ensuite, on les associe avec leur propre symbole.

Après cela, on vérifie les scores des deux joueurs, le premier joueur avec un score supérieur ou égal à 10 sera automatiquement déclaré vainqueur de la partie puisqu'on mettra la variable fini à 1.

Ensuite, on met l'entrée utilisateur. On demande à l'utilisateur (ici le joueur actif) de saisir l'indice du pion qu'il veut placer. On lui rappelle également de saisir l'indice 0,0 s'il veut abandonner.

Pour gérer l'abandon, on détecte si la ligne et la colonne saisie (ce sont les variables locales de l'action) sont égales à 0. Si c'est le cas, la variable fini sera égale à 1, l'adversaire sera le vainqueur et le joueur actif sera le perdant.

On doit aussi vérifier si l'utilisateur n'a pas saisi un indice en dehors du plateau: On vérifie si la ligne et la colonne ne sont pas en dehors du plateau et on vérifie si l'indice saisi ne contient pas de '.' . On vérifiera une autre fois si la ligne ou la colonne saisie est différente de 0 pour ne pas afficher le message "Veuillez réessayer" quand l'indice saisi est (0,0). Si l'entrée n'est pas valide, l'utilisateur va devoir réessayer. Sinon, on placera le symbole du joueur actif dans la case d'indice saisi par l'utilisateur, puis on appellera l'action verif_captures pour détecter si on peut effectuer une capture à partir du pion saisi.

Après cela, on détecte s'il y a un alignement en appelant la fonction verif_alignement. Si on retrouve un alignement, on affiche le plateau une dernière fois et on enverra un message pour annoncer la victoire du joueur actif et enfin on mettra la variable fini à 1 pour finir la partie.

Enfin, on va alterner les tours en multipliant la variable tour par -1. Pour passer de O à X, on fait (-1) *(-1) = 1 (ce qui est bien la valeur associée au symbole X). Pour passer de X à O, on fait 1*(-1) = -1 (valeur associée au symbole O).

4.5- int main() (fonction principale)

```c
/*************************************************************/
/* Fonction principale                                       */
/* fini : Entier - Permet de finir la partie - Locale        */
/*************************************************************/
int main()
{
    Jeu pente;
    init_jeu(&pente);

    int fini = 0;
    while (!fini)
    {
        affichage_plateau(pente);
        tour_joueur(&pente, &fini);
    }

    return 0;
}
```

La fonction principale commence par une initialisation d'une structure de type Jeu qui s'appellera pente. Cette structure de type Jeu contiendra toutes les variables mentionnées précédemment dans la description et la justification de la structure de données (partie 3.2).

Ensuite, on va appeler l'action init_jeu qui prend en entrée l'adresse de la structure qu'on vient d'initialiser.

Après, on va initialiser une variable locale appelée 'fini' à 0. Cette variable locale permettra d'arrêter la partie: Si 'fini' est égale à 0, la partie continue. Sinon (ou si 'fini' est égale à 1), la partie est finie.

Pour pouvoir appliquer ce qu'on vient d'expliquer sur la variable 'fini', on doit faire une boucle qui s'exécute tant que 'fini' est égale à 0. Dans cette boucle, on affiche l'état actuel du plateau et les scores en appelant l'action affichage_plateau qui prend en entrée la structure 'pente' mise à jour bien évidemment. Ensuite, on appelle l'action tour_joueur pour faire toutes les vérifications nécessaires pour chaque tour. Cette action prend en entrée l'adresse de la structure 'pente', ainsi que l'adresse de la variable 'fini' (pour la mettre à 1 si une condition de victoire est détectée, sinon elle reste à 0 et on continue à exécuter la même boucle).

Et enfin, on met un return 0 après la boucle while pour que le programme soit terminé.

5- Progrès du projet et défis rencontrés

Séance 1 (07/01/2025):

Lors de notre première séance en projet d'algorithmique et programmation, nous avons pris le temps d'explorer en profondeur le sujet qui nous a été donné afin d'en comprendre tous les aspects de manière détaillée. Cette étape préliminaire nous a permis de poser des bases solides avant de commencer la conception du programme. On a commencé alors par rédiger un cahier des charges précis, en identifiant clairement les objectifs, les contraintes et les limites de ce projet. On a également effectué une analyse approfondie du problème, ce qui s'est révélé particulièrement bénéfique pour la suite, puisque cette démarche nous a aidé à structurer et répartir efficacement les différents algorithmes nécessaires à la réalisation du projet.

En outre, on a accordé une attention particulière à l'optimisation des solutions envisagées, puisqu'on cherchait à concevoir un programme qui était à la fois performant et facile à comprendre une fois finalisé. Donc même si on a eu quelques problèmes au niveau de la créativité au départ, cette séance méthodique nous a facilité les étapes suivantes du développement du projet.

Séance 2 (09/01/2025):

Pour la deuxième séance, nous avons totalement focalisé notre attention sur la phase de programmation. Après avoir établi une répartition claire des tâches, chacun a fait la conception, ainsi que le développement d'un sous programme spécifique de manière individuelle. Chaque partie était testée individuellement pour s'assurer de son bon fonctionnement avant de les intégrer progressivement dans un même programme.

Bien que les débuts aient été marqués par diverses difficultés d'intégration, nous avons réussi, au fil de la séance, à assembler plusieurs sous-programmes qui nous ont aidés à avancer. Alors, nous avons démarré par des fonctions/actions relativement simples, telles que l'affichage et l'initialisation du plateau, ce qui nous a permis déjà de construire une base solide.

Par ailleurs, nous avons également commencé à travailler sur la fonction de vérification d'alignement de pions, qui était un peu difficile au départ, mais éventuellement, a été finalisée vers la fin de la séance. En revanche, la rédaction du sous-programme pour les prises de pions s'est avérée être un défi complexe. Cette partie du projet nous a demandé un effort considérable et c'est cette partie-là qui nous a pris le plus de temps pendant ce projet. Malheureusement, on a pas pu finir cette partie dans la séance 2. Mais, cette séance a marqué une étape clé dans notre progression.

Séance 3 (15/01/2025, 10h15/12h15):

C’est pendant cette séance que notre projet commençait à prendre forme. Alors, nous nous sommes d’abord concentrés sur le sous-programme dédié à la gestion des prises, une partie cruciale qui nous empêchait de poursuivre le développement du programme. Après environ une heure de travail, nous avons finalement trouvé une solution efficace et fonctionnelle avec les tests effectués. On a donc pu progresser significativement lors de cette séance.

On a enchaîné avec la création d’un sous-programme central qui gère les différentes fonctions et actions dans le programme avec des vérifications. C’est également au cours de cette séance que nous avons mis en place un mécanisme simple et efficace pour alterner les joueurs à chaque tour, qui était une étape cruciale pour la conception du programme.

Une fois que tous les sous-programmes ont été mis en place, on a rédigé le programme principal. Cette étape était plutôt simple, puisque les vérifications des conditions de victoire et l’alternance des tours avaient déjà été dans le sous-programme qui vient d’être mentionné dans le paragraphe dernier. Donc il nous suffisait d’implémenter une boucle while pour répéter le même déroulement du jeu jusqu'à ce qu’une condition de victoire soit détectée.

Nous avons donc pu finaliser le programme avant la fin de la séance. Pour en profiter, nous avons commencé à rédiger le rapport ainsi que les fichiers tests.

Séance 4 (15/01/2025, 13h45, 15h45):

Lors de la quatrième et dernière séance, on s’est principalement concentré sur la rédaction du rapport, ainsi que la vérification de la fonctionnalité du programme avec nos propres fichiers tests. Tous nos fichiers tests ont été exécutés pour vérifier le fonctionnement du programme, et les résultats obtenus étaient cohérents (les fichiers tests vont être présentés dans la partie suivante où on donnera le résultat attendu de chaque fichier test).

Concernant le rapport, on a bien avancé dans la rédaction de plusieurs parties, qui sont les suivantes :

- Introduction: On s’est contenté de présenter le concept du jeu avec toutes les règles à connaître.
- Cahier des charges: On a donné les objectifs du projet, accompagnés par une analyse approfondie et la présentation des contraintes et limites.
- Analyse du problème: On a étudié les défis principaux du projet avec une méthode pour résoudre cela.
- Algorithme des captures: On s’est inspiré des anciens algorithmes faits en TD pour traduire notre sous-programme C en écriture algorithmique.

6- Fichiers tests

Dans cette partie, on va vous présenter les fichiers tests qu'on a rédigé pour couvrir le plus de scénarios possibles afin de valider le fonctionnement du programme. Les voici:

pente3.txt:

```
13,5
13,6
12,6
12,7
11,7
11,8
10,8
11,11
9,9
10,10
10,6
10,5
11,9
11,10
12,10
7,10
```

```
Joueur de symbole O: Indiquez l'indice (ligne, colonne) où vous voulez placer votre pion (entrez 0,0 pour abandonner).
      1         2         3         4         5         6         7         8         9         10        11        12
 1    .         .         .         .         .         .         .         .         .         .         .         .
 2    .         .         .         .         .         .         .         .         .         .         .         .
 3    .         .         .         .         .         .         .         .         .         .         .         .
 4    .         .         .         .         .         .         .         .         .         .         .         .
 5    .         .         .         .         .         .         .         .         .         .         .         .
 6    .         .         .         .         .         .         .         .         .         .         .         .
 7    .         .         .         .         .         .         .         .         .         .         .         .
 8    .         .         .         .         .         .         .         .         .         .         .         .
 9    .         .         .         .         .         .         .         .         O         .         .         .
10    .         .         .         .         .         .         .         O         .         .         .         .
11    .         .         .         .         .         .         O         X         .         .         X         .
12    .         .         .         .         .         O         X         .         .         .         .         .
13    .         .         .         .         O         X         .         .         .         .         .         .
14    .         .         .         .         .         .         .         .         .         .         .         .
15    .         .         .         .         .         .         .         .         .         .         .         .
16    .         .         .         .         .         .         .         .         .         .         .         .
17    .         .         .         .         .         .         .         .         .         .         .         .
18    .         .         .         .         .         .         .         .         .         .         .         .
19    .         .         .         .         .         .         .         .         .         .         .         .
Nombre de pions capturés:
X: 0          O: 0
Le joueur portant le symbole O a gagné par alignement
```

Résultat attendu: Victoire de O par alignement diagonal ascendant

pente4.txt:

```
9,9
9,10
10,9
10,10
11,9
11,10
12,9
11,11
10,7
10,10
10,6
10,5
11,9
11,10
12,10
7,10
```

```
Joueur de symbole O: Indiquez l'indice (ligne, colonne) où vous voulez placer votre pion (entrez 0,0 pour abandonner).
      1         2         3         4         5         6         7         8         9         10        11        12        1
 1    .         .         .         .         .         .         .         .         .         .         .         .
 2    .         .         .         .         .         .         .         .         .         .         .         .
 3    .         .         .         .         .         .         .         .         .         .         .         .
 4    .         .         .         .         .         .         .         .         .         .         .         .
 5    .         .         .         .         .         .         .         .         .         .         .         .
 6    .         .         .         .         .         .         .         .         .         .         .         .
 7    .         .         .         .         .         .         .         .         .         .         .         .
 8    .         .         .         .         .         .         .         .         .         .         .         .
 9    .         .         .         .         .         .         .         .         O         X         .         .
10    .         .         .         .         .         .         .         .         O         X         .         .
11    .         .         .         .         .         .         .         .         O         X         .         .
12    .         .         .         .         .         .         .         .         O         .         .         .
13    .         .         .         .         .         .         .         .         .         .         .         .
14    .         .         .         .         .         .         .         .         .         .         .         .
15    .         .         .         .         .         .         .         .         .         .         .         .
16    .         .         .         .         .         .         .         .         .         .         .         .
17    .         .         .         .         .         .         .         .         .         .         .         .
18    .         .         .         .         .         .         .         .         .         .         .         .
19    .         .         .         .         .         .         .         .         .         .         .         .
Nombre de pions capturés:
X: 0          O: 0
Le joueur portant le symbole O a gagné par alignement
```

Résultat attendu: Victoire de O par alignement vertical

pente5.txt:

```
9,9
10,9
9,10
10,10
9,11
10,11
9,12
11,11
9,13
10,10
10,6
10,5
11,9
11,10
12,10
7,10
```

```
Joueur de symbole O: Indiquez l'indice (ligne, colonne) où vous voulez placer votre pion (entrez 0,0 pour abandonner).
      1         2         3         4         5         6         7         8         9         10        11        12        13        14
 1    .         .         .         .         .         .         .         .         .         .         .         .         .         .
 2    .         .         .         .         .         .         .         .         .         .         .         .         .         .
 3    .         .         .         .         .         .         .         .         .         .         .         .         .         .
 4    .         .         .         .         .         .         .         .         .         .         .         .         .         .
 5    .         .         .         .         .         .         .         .         .         .         .         .         .         .
 6    .         .         .         .         .         .         .         .         .         .         .         .         .         .
 7    .         .         .         .         .         .         .         .         .         .         .         .         .         .
 8    .         .         .         .         .         .         .         .         .         .         .         .         .         .
 9    .         .         .         .         .         .         .         .         O         O         O         O         O         .
10    .         .         .         .         .         .         .         .         X         X         X         .         .         .
11    .         .         .         .         .         .         .         .         .         .         X         .         .         .
12    .         .         .         .         .         .         .         .         .         .         .         .         .         .
13    .         .         .         .         .         .         .         .         .         .         .         .         .         .
14    .         .         .         .         .         .         .         .         .         .         .         .         .         .
15    .         .         .         .         .         .         .         .         .         .         .         .         .         .
16    .         .         .         .         .         .         .         .         .         .         .         .         .         .
17    .         .         .         .         .         .         .         .         .         .         .         .         .         .
18    .         .         .         .         .         .         .         .         .         .         .         .         .         .
19    .         .         .         .         .         .         .         .         .         .         .         .         .         .
Nombre de pions capturés:
X: 0           O: 0
Le joueur portant le symbole O a gagné par alignement
```

Résultat attendu: Victoire de O par alignement horizontal

pente6.txt:

```
5,5
5,6
6,6
6,7
7,7
7,8
8,8
11,11
9,9
10,10
10,6
10,5
11,9
11,10
12,10
7,10
```

```
Joueur de symbole O: Indiquez l'indice (ligne, colonne) où vous voulez placer votre pion (entrez 0,0 pour abandonner).
      1         2         3         4         5         6         7         8         9         10        11        12
 1    .         .         .         .         .         .         .         .         .         .         .         .
 2    .         .         .         .         .         .         .         .         .         .         .         .
 3    .         .         .         .         .         .         .         .         .         .         .         .
 4    .         .         .         .         .         .         .         .         .         .         .         .
 5    .         .         .         .         O         X         .         .         .         .         .         .
 6    .         .         .         .         .         O         X         .         .         .         .         .
 7    .         .         .         .         .         .         O         X         .         .         .         .
 8    .         .         .         .         .         .         .         O         .         .         .         .
 9    .         .         .         .         .         .         .         .         O         .         .         .
10    .         .         .         .         .         .         .         .         .         .         .         .
11    .         .         .         .         .         .         .         .         .         .         X         .
12    .         .         .         .         .         .         .         .         .         .         .         .
13    .         .         .         .         .         .         .         .         .         .         .         .
14    .         .         .         .         .         .         .         .         .         .         .         .
15    .         .         .         .         .         .         .         .         .         .         .         .
16    .         .         .         .         .         .         .         .         .         .         .         .
17    .         .         .         .         .         .         .         .         .         .         .         .
18    .         .         .         .         .         .         .         .         .         .         .         .
19    .         .         .         .         .         .         .         .         .         .         .         .
Nombre de pions capturés:
X: 0           O: 0
Le joueur portant le symbole O a gagné par alignement
```

Résultat attendu: Victoire de O par alignement diagonal descendant

pente7.txt:

```
9,9
9,8
12,12
9,7
9,6
13,12
16,17
14,12
15,12
15,16
18,1
14,15
13,14
17,2
3,3
16,3
15,4
15,5
11,4
15,6
15,7
17,15
8,6
9,7
7,6
6,6
```

```
    1   2   3   4   5   6   7   8   9  10  11  12  13
 1  .   .   .   .   .   .   .   .   .   .   .   .   .
 2  .   .   .   .   .   .   .   .   .   .   .   .   .
 3  .   .   0   .   .   .   .   .   .   .   .   .   .
 4  .   .   .   .   .   .   .   .   .   .   .   .   .
 5  .   .   .   .   .   .   .   .   .   .   .   .   .
 6  .   .   .   .   .   .   .   .   .   .   .   .   .
 7  .   .   .   .   .   .   .   .   .   .   .   .   .
 8  .   .   .   .   .   .   .   .   .   .   .   .   .
 9  .   .   .   .   .   0   .   .   0   .   .   .   .
10  .   .   .   .   .   .   .   .   .   .   .   .   .
11  .   .   .   0   .   .   .   .   .   .   .   .   .
12  .   .   .   .   .   .   .   .   .   .   .   0   .
13  .   .   .   .   .   .   .   .   .   .   .   .   .
14  .   .   .   .   .   .   .   .   .   .   .   .   .
15  .   .   .   0   .   .   0   .   .   .   .   0   .
16  .   .   .   .   .   .   .   .   .   .   .   .   .
17  .   .   .   .   .   .   .   .   .   .   .   .   .
18  0   .   .   .   .   .   .   .   .   .   .   .   .
19  .   .   .   .   .   .   .   .   .   .   .   .   .
Nombre de pions captures:
X: 0           O: 10
Joueur O gagne par captures
```

Résultat attendu: Victoire de O par captures (10 prises)

Ces fichiers tests ont été ajoutés dans le fichier zip avec le programme et les fichiers tests déjà donnés (pente1.txt et pente2.txt).

7- Conclusion

Ce projet a consisté à concevoir et développer un programme en C qui nous permet de jouer au jeu de “Penté”, un jeu stratégique qui possède des règles spécifiques. À travers une modélisation efficace de la structure du programme pour le jeu, nous avons pu implémenter les fonctionnalités essentielles: alignement, captures, gestion des tours et abandon. On a également effectué l’affichage dynamique du plateau avec une validation des entrées utilisateur pour avoir une expérience optimale.

Malgré les nombreux défis rencontrés, comme la rédaction du sous-programme des prises dans toutes les directions, ces obstacles ont été surmontés grâce à un travail en équipe, un travail qui a été précis, rigoureux et avec beaucoup d’efforts. Enfin, on a réussi à compiler le programme sans erreur, avec tous nos fichiers tests fonctionnels, affichant bien le résultat attendu.

Ce projet a permis de renforcer nos compétences en algorithmique, tel que la conception d’un programme, le fait d’analyser un programme et retrouver ses erreurs, ainsi que l’optimiser. Ca a aussi renforcé nos compétences en travail collaboratif, ce qui était un aspect inédit lors de ce module. Donc, on a réussi à faire un programme répondant au cahier des charges, mais qui peut contenir d'éventuelles améliorations pour une meilleure expérience utilisateur comme l’ajout d’une interface graphique, l’ajout d’une option multijoueur en ligne ou aussi l’ajout d’une sauvegarde et reprise des parties.