Crémetz Olivier GIS 1

Mssiaidi Ali Groupe 2

TP PENTE

COMPTE-RENDU

Polytech'Lille – Département Génie Informatique et Statistique
Année 2004-2005

SOMMAIRE

1. Introduction 3
2. Présentation des règles du jeu 3
3. Objet de l'étude 4
4. Analyse du problème 4
5. Description de la structure de données utilisée 6
6. Description des principaux algorithmes 7
7. Compilation et utilisation du programme 14
8. Listing du programme 15
9. Conclusion 24

1. Introduction

Dans le cadre des travaux pratiques, nous devons concevoir et implémenter le jeu du Penté. Nous allons donc vous présenter les règles du jeu et notre façon de le créer.

2. Présentation des règles du jeu

But du jeu

Le joueur qui réalise l'un des deux objectifs suivants est déclaré vainqueur immédiatement:

- Aligner cinq pions contigus de sa couleur horizontalement, verticalement ou en diagonale. On appelle cet alignement un "Pente".
- Avoir pris cinq paires de pions adverses.

Attention: il suffit de réaliser l'un *ou* l'autre de ces objectifs pour gagner immédiatement. Cette double possibilité de victoire constitue l'un des attraits du jeu.

Matériel

Un plateau de 19 cases sur 19. Chaque joueur dispose de pions de sa couleur (X ou O).

Principe et tour de jeu

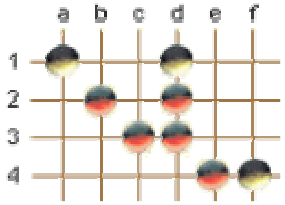
Le joueur qui commence est celui qui possède les ronds O. Il place un de ses pions sur une intersection du plateau. Ensuite, à tour de rôle, les joueurs posent un de leurs pions sur une intersection libre du plateau.

Prise de pions

Lorsqu'un joueur pose un pion qui prend en tenaille (comme à l'Othello) une *paire* adverse entre deux de ses pions, la paire adverse est retirée. La tenaille peut se faire horizontalement, verticalement ou en diagonales. On ne peut prendre que des *paires* de pions adverses, jamais un pion seul ou plus de deux pions.

Attention: quand un joueur vient de prendre la cinquième paire de pions adverse, il gagne immédiatement la partie!

Un seul pion peut réaliser plusieurs prises d'un coup. Jaune joue en d4 et prend deux paires de pions rouges (b2, c3, d2 et d3):



Un joueur peut poser un pion à l'intérieur d'une tenaille sans risque. Jaune joue en b2 et ne perd pas de pion:

3. Objet de l'étude

Le but de ce projet est de réaliser un jeu dans sa totalité, c'est à dire de l'idée à l'exécutable.
De ce fait, nous allons apprendre à gérer de nombreux paramètres :
Le premier est évident et découle de la signification même du nom commun projet : se projeter dans le futur. Donc il faudra anticiper et savoir par avance ce que l'on sera capable de faire.
Le deuxième est d'apprendre à gérer le temps : calendrier de remise du tp.
Le troisième est d'apprendre à travailler de manière collective : intégrer un travail réalisé individuellement dans un projet collectif.
Donc l'intérêt de notre projet est clair :

- Ouverture à la programmation
- Découverte d'une situation de travail analogue à celle rencontrée en entreprise
- Jouer à notre propre jeu

4. Analyse du problème

La difficulté du sujet réside dans l'identification des algorithmes pour la construction du jeu proprement dit.
La conception du jeu Pente pourra se découper de la façon suivante :

- Initialisation de la grille
- Affichage de la grille de jeu
- Saisie d'un coup
- Vérification si un alignement continu de 5 pions a été réalisé
- Chercher à réaliser les prises
- Vérifier si la grille est pleine
- Afficher le gagnant

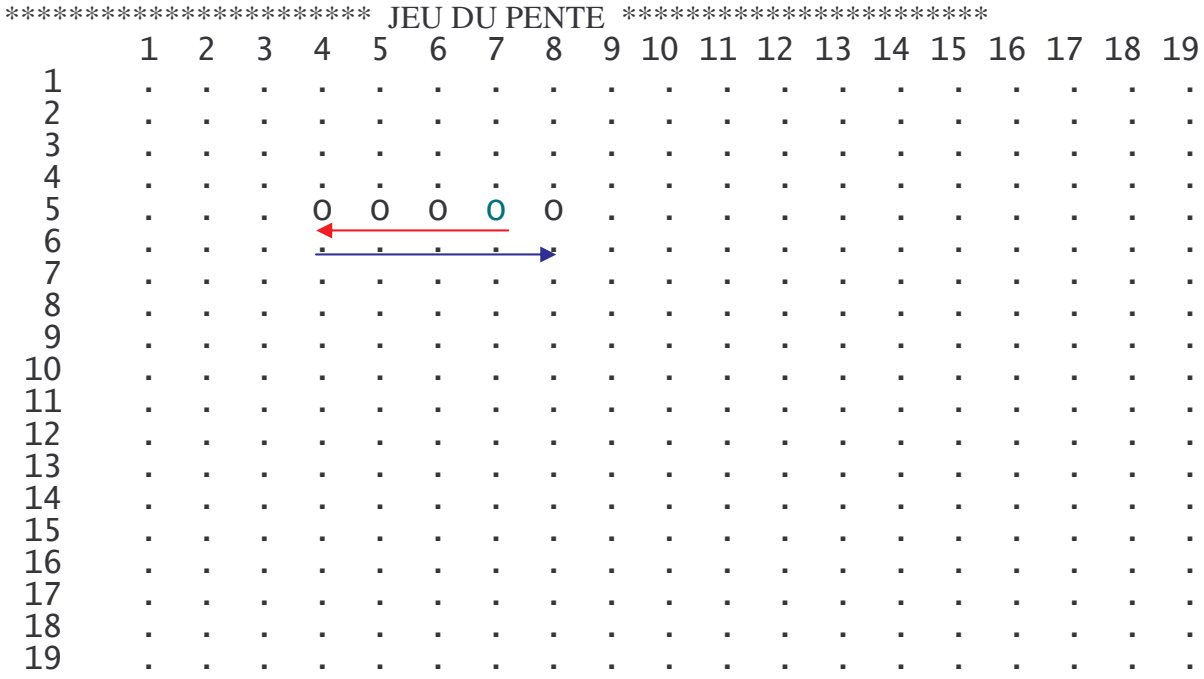
Le but sera de minimiser les calculs
Le joueur saisira d'abord la ligne puis la colonne. A chaque saisie, la grille sera réaffichée et les tests se feront automatiquement.
Tant que la grille n'est pas pleine, le jeu continue. Il s'arrête dès lors qu'un joueur a réussi à prendre les 10 pions de son adversaire ou qu'il ait réussi à aligner 5 pions en continu.
Les conditions ainsi que la taille de la grille seront mis en constantes au début du programme si on désire changer les règles du jeu (Cela évite de rechercher tous les endroits où ces règles sont inscrites).

Pour des raisons de simplicité dans la programmation, nous avons considéré que la matrice (représentant la grille) était carrée. La plupart du temps cette grille est de taille 19*19, mais il peut arriver que l'on désire jouer avec une grille 9*9 ou 13*13.
Il y aura aussi que 2 joueurs car les tests d'entourage de pions serait plus difficile à mettre en place.
Le joueur 1 prendra automatiquement le pion O et démarrera la partie comme indiqué sur le sujet. Le joueur 2 aura donc les pions X.
Nous n'avons pas limité le nombre de pions ce qui permet d'atteindre le maximum de la grille.
Pour les tests nous avons décidé de dédier 4 constantes correspondant à l'alignement :

- en ligne
- en colonne
- sur la diagonale montante
- sur la diagonale descendante

Une procédure nous renverra le déplacement en ligne et colonne pour atteindre respectivement le coté gauche, le haut, le coin bas gauche et le coin haut gauche.
Cela servira à éviter de coder 4 fonctions différentes pour les tests.
Dans le test d'alignement nous regardons si on ne retrouve pas directement les 5 pions alignés en partant dans la direction donnée par la procédure. Si nous ne retrouvons pas le bon résultat, nous partons du coin le plus à gauche pour compter le nombre de pions alignés dans l'autre sens

Exemple :
Test en ligne
Nous venons de joueur en ligne 5 colonne 7

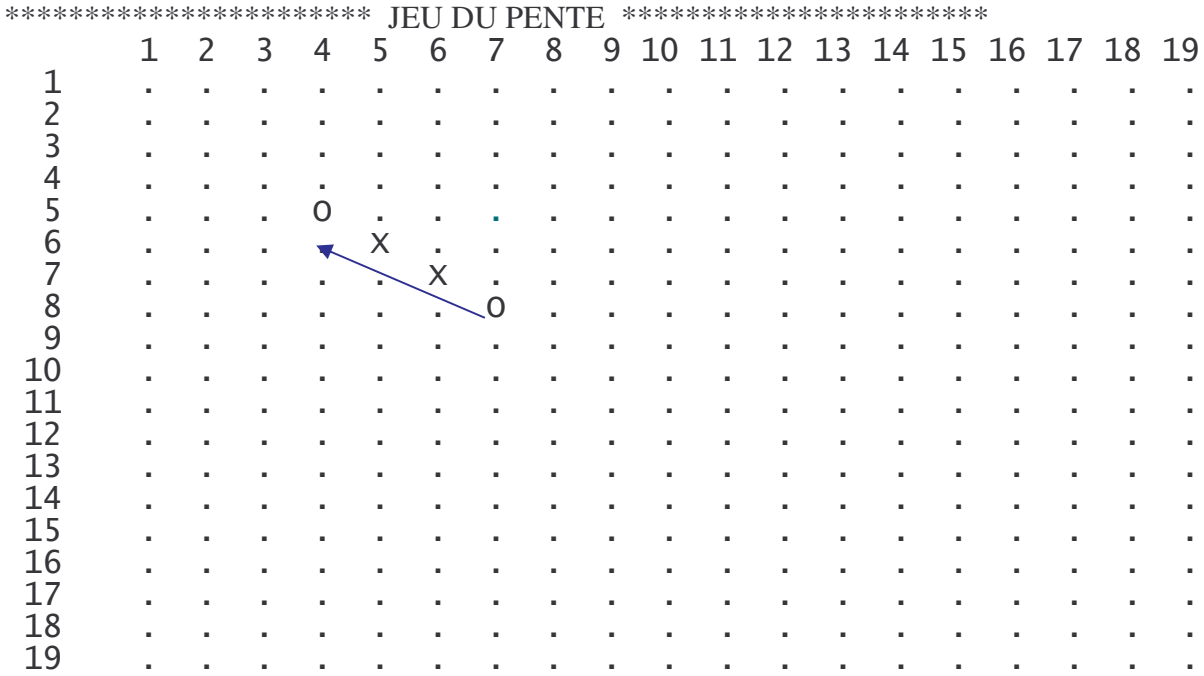


1ère phase, on recherche si on a les 5 pions alignés dans ce sens (Ici non)
2ème phase, on recherche de la position la plus à gauche et on compte les pions qui nous appartiennent et qui sont juste continu
Ici le joueur a réalisé un alignement

Si on avait le résultat dès la première phase, le joueur aurait été déclaré vainqueur tout de suite. Le test se déroule en 2 phases pour éviter de réaliser des opérations mathématiques sur le compteur à partir de la position trouvée.

Pour l'entourage on cherche dans tout les 8 sens : (Haut, bas, gauche, droite, haut gauche, bas gauche, haut droit et bas droit).
On se base sur la procédure de déplacement qui nous donne l'accès au coté gauche puis pour les 4 autres cotés, on inverse le sens de déplacement

Exemple de prise de pions :



Exemple le joueur O vient de jouer en position 8,7 : Il vient d'entourer 2 pions X de son adversaire. Les pions X vont donc disparaître et le compteur de prise de pion du joueur va s'incrémenter de 2

5. Description de la structure de données utilisée

Le langage de programmation utilisé est le langage C de l'environnement LINUX.

Dans ce programme nous allons mettre en place une structure joueur définie comme suivant :

```
Type Joueur = Structure
  Num :entier
  Prenom : chaine de caractere de taille[20];
  Pion : caractere
  Nbpionsadv : entier
fin
```

Le joueur sera donc caractérisé par un numéro, son prénom, son pion O ou X et le nombre de pions pris à l'adversaire
Grâce à la structure nous pourrons connaître le pion d'un joueur et de savoir son nombre de pions pris à l'adversaire lors du test d'entourage ou lorsque l'on teste si le joueur a gagné.
La grille de jeu sera représentée par une matrice carrée de taille 19*19. Nous avons choisi cette structure car la taille de la grille est fixe et on a pas le besoin d'ajouter de nouvelles lignes ou colonnes durant la partie. L'accès à un élément du tableau se fera grâce à ses coordonnées dans la matrice.

6. Description des principaux algorithmes

Algorithme d'initialisation de la ligne :
On parcours la grille et on met un caractère espace dans caque case du tableau.

```
Action : initGrille(grille)
D :
D/R : grille matrice carrée de caractères de [0..19][0..19]
R :
L : ligne,colonne : entier {indice de boucles}

Pour ligne de 0 à 19 faire
      Pour colonne de 0 à 19 faire
            grille[ligne][colonne]=' ';
       Fait
Fait
Fin Action
```

Affichage de la grille :
On affiche toute la grille

```
Action : afficheGrille(grille)
D :
D/R : grille matrice carrée de caractères de [0..18][0..18]
R :
L : ligne,colonne : entier {indice de boucles}
{On affiche les numeros de colonnes}
  Afficher("     ");
  Pour colonne de 0 à 18 faire
      Afficher ( colonne+1 );
  Fait
 Afficher(saut de ligne) { on passe une ligne}

{On affiche pour chaque ligne le numéro de la ligne puis les pions}
  Pour ligne de 0 à 18 faire
     Afficher (ligne+1);
      Pour colonne de 0 à 18 faire
            Si grille[ligne][colonne] = 'X' ou grille[ligne][colonne] ='O' Alors
              Afficher( grille[ligne][colonne] );
            Sinon
```

```
            Afficher(«  . »);
          Fin Si
      Fin Pour
     Afficher(saut de ligne) { on passe une ligne}
 Fin Pour
Fin Action
```

Saisie d'un coup sur la grille :

On demande à l'utilisateur de saisir la ligne et la colonne. Si le coup est déjà pris l'utilisateur devra resaisir. On se sert ici de la structure joueur pour connaître le pion du joueur en cours.

```
Action : saisie(grille, joueurCourant, lignesaisie, colonnesaisie)
D : joueurCourant de type Joueur
D/R : grille matrice carrée de caractères de [0..18][0..18]
R : lignesaisie et colonnesaisie : entier
L :  Recommencer : booléen
{init}
  Recommencer := vrai

  Tant que (recommencer) faire
    Lire (« Entrer la ligne », lignesaisie)
    Lire (« Entrer la colonne », colonnesaisie)

    { Si les coordonnees sont en dehors du tableau ou si la place n'est pas libre, on recommence
la saisie }
    Si (lignesaisie>=1 et lignesaisie<=19 et colonnesaisie>=1 et colonnesaisie<=19 et
grille[lignesaisie-1][colonnesaisie-1]!='O' et grille[lignesaisie-1][colonnesaisie-1]!='X') alors
          recommencer=FAUX;
    Sinon
         Afficher("Saisie incorrecte veuillez recommencer\n");
         recommencer=VRAI;
    Fin si
  Fait
  { On affiche sur la grille la nouvelle position}
  grille[lignesaisie-1][colonnesaisie-1]=joueurCourant.pion;
fin action
```

Vérification si la grille est pleine :

Cette fonction parcourt la grille. Dès qu'elle ne rencontre aucun des 2 pions de la partie, elle renvoie faux

```
Fonction : grillePleine(grille)
D : grille matrice carrée de caractères de [0..18][0..18]
L : ligne,colonne : entier {indice de boucles}, res : booleen
{init}
res :=VRAI
```

```
   Pour i de 0 à 18 faire
    Pour j de 0 à 18 faire
          { Si une position de la grille ne contient pas les pions X et O on s'arrete   }
          Si (grille[i][j]!='O' et grille[i][j]!='X') Alors
            res := FAUX
          Fin si
         Fait
  Fait
  Resultat := res
Fin Fonction
```

Vérification si un joueur a gagné la partie : (Alignement de 5 pions)

Pour vérifier qu'un joueur a réussi à aligner ses 5 pions, nous mettons en place notre méthode citée dans l'analyse du problème. Pour cela nous avons besoin d'une procédure permettant de renvoyer le point de départ pour tracer la droite :

en ligne : à gauche	diagonale montante : coin bas gauche
colonne : en haut	diagonale descendante : coin haut gauche

Nous avons à disposition les constantes de directions suivantes :

LIGNE := 1	DIAG_MONTANTE := 3
COLONNE := 2	DIAG_DESCENDANTE :=4

```
Action : deplacement(direction, deplacLigne, deplacColonne)
D : direction de type entier (voir les constantes)
D/R :
R : deplacLigne,deplacColonne : entier
L :
  Si (direction = LIGNE) Alors
     { En ligne on part vers la gauche }
      deplacLigne := 0
      deplacColonne :=-1
  Fin Si
  Si (direction = COLONNE) Alors
    { En Colonne on part vers le haut }
      deplacLigne := -1
      deplacColonne := 0
  Fin Si
  Si (direction = DIAG_MONTANTE) Alors
    { En Diagonale montante on part vers le bas gauche }
      deplacLigne := 1
      deplacColonne := -1
   Fin Si
   Si (direction = DIAG_DESCENDANTE) Alors
    { En Diagonale descendante on part vers le haut gauche }
      deplacLigne := -1
      deplacColonne := -1
    Fin Si
Fin Action
```

```
Fonction : testJeuGagne(grille, lignesaisie, colonnesaisie, direction) : booléen
D : direction de type entier (voir les constantes)
      lignesaisie, colonnesaisie : entier { dernier coup saisi}
      grille matrice carrée de caractères de [0..18][0..18]
L : compteur, compteurtemporaire : entier { compteur des pions alignés }
     Res : booléen
     i,j : entier {dernier coup saisi transposé à la grille de 0 à 18)
     pionatester :caractère {pion à tester}
    deplacLigne,deplacColonne : entier {déplacement à effectuer pour aller vers la gauche}
{init}
compteur=0;
compteurtemporaire=0;
pion_a_tester=grille[lignesaisie-1][colonnesaisie-1];
{ on transpose le coup sur la matrice }
 i=lignesaisie-1
 j=colonnesaisie-1
 b :=faux
{ On récupère le déplacement)
 deplacement(direction, deplacLigne, deplacColonne)

 {On déplace dans la première direction (jusqu'au premier pion appartenant au joueur)}
 {Quand on se déplace par la gauche on n'a pas besoin de teste si la colonne touche le cote
droit}
tant que (i>=0 et j>=0 et i<TAILLE et grille[i+deplacLigne][j+deplacColonne] =
pion_a_tester) faire
     i=i+deplacLigne
     j=j+deplacColonne
     { On compte au cas ou on aurait tout dans le sens inverse}
     compteurtemporaire=compteurtemporaire+1
 Fait

 {Si on a déjà le compte, on retourne directement le résultat on ne rentre pas dans la 2eme
boucle}
 Si (compteurtemporaire = 10) Alors
   Res := VRAI
 Fin Si
 { On va dans l'autre sens : Quand on se déplace par la droite on n'a pas besoin de teste si la
colonne touche le cote gauche}
 Tant que ( !Res et compteur!=10 et grille[i][j]==pion_a_tester et i>=0 et i<19 et j<19) Faire
    compteur=compteur+1
     i=i-deplacLigne
     j=j-deplacColonne
   Fait
 { Si on a le compte, on a gagne}
 Si (compteur = 10) Alors
   Res := VRAI;
 Fin Si
Resultat := Res
Fin fonction
```

On lance le test précédent sur les 4 directions.
Fonction : testGagne(grille, lignesaisie, colonnesaisie) : booléen
D : lignesaisie, colonnesaisie : entier { dernier coup saisi}
grille matrice carrée de caractères de [0..18][0..18]
L : Res : booléen
i : entier {indice de parcours)
{init}
Res :=faux
i=1;
{ On test les 4 directions}
Tant que(i<=4 et !testJeuGagne(grille,lignesaisie,colonnesaisie,i)) Faire
i=i+1;
Fait
{ Si la boucle s'est arrêtée avant de faire toutes les directions, on a gagne}
Si(i<=4) Alors
Res:= Vrai;
Fin si
Résultat := Res
Fin Fonction

Vérification si un joueur a gagné la partie : (Prise de 10 pions)

Action : testEntoure (grille, joueurCourant, lignesaisie, colonnesaisie)
D : lignesaisie et colonnesaisie : entier
D/R : grille matrice carrée de caractères de [0..18][0..18]
joueurCourant,j1,j2 : de type joueur.{La structure permet de récuperer son pion, son nombre}
R :
L : i,j : entier{dernier coup saisi transposé à la grille de 0 à 18), pion, pionentoure : caractère deplacLigne,deplacColonne{coordonnées du point de départ},sens {sens pour le déplacement}: entier
M,n :entier { indice de boucle}
{On transpose les coordonnées représentant le dernier coup joue par le joueur j dans la grille}
i=lignesaisie-1
j=colonnesaisie-1

{On récupère le pion du joueur courant}
pion=joueurCourant->pion

{On récupère le pion de l'adversaire}
Si (pion = 'O') Alors
pionentoure='X'
Sinon
pionentoure='O'
Fin Si

{on teste dans les 4 directions}
Pour m de 1 à 4 Faire
{On teste dans la direction des deux cotes}
Pour n de 0 à 1 Faire

```
            Si (n = 0) Alors
              sens=1
            Sinon
              Sens = -1 {On inverse le sens}
            Fin Sinon
            //On recupere le sens du deplacement
            deplacement(m, &deplacLigne,&deplacColonne);

           Si (i+3*sens*deplacLigne >= 0 et i+3*sens*deplacLigne <= TAILLE et
j+3*sens*deplacColonne>= 0 et j+3*sens*deplacColonne <= TAILLE et
(grille[i+sens*deplacLigne*1][j+sens*deplacColonne*1]==pionentoure) et
(grille[i+sens*deplacLigne*2][j+sens*deplacColonne*2]==pionentoure) et
(grille[i+sens*deplacLigne*3][j+sens*deplacColonne*3]==pion)) Alors
              { On incremente le nombre de pions du joueur Courant et celui du joueur}
              Si (joueurCourant.num = j1.num) Alors
                j1.nbpionsadv=j1.nbpionsadv+2 {On incrémente les pions des adversaires de j1}
                joueurCourant.nbpionsadv=joueurCourant.nbpionsadv+2
              Sinon
                j2.nbpionsadv=j2.nbpionsadv+2{On incrémente les pions des adversaires de j2}
                joueurCourant.nbpionsadv=joueurCourant.nbpionsadv+2
              Fin Si

              { On supprime la paire de pion que l'on a entoure}
              supprimePion(grille,i+sens*deplacLigne*1,j+sens*deplacColonne*1)
              supprimePion(grille,i+sens*deplacLigne*2,j+sens*deplacColonne*2)
              afficheGrille(grille)
            Fin Si
          Fin Pour
    Fin Pour
Fin action


Action : supprimePion (grille,coordV, coordH)
D : coordV, coordH : entier {coordonnée du pion à supprimer}
D/R : grille matrice carrée de caractères de [0..18][0..18]
R :
  grille[coordV][coordH]=' ';
Fin Action
```

Logique du jeu

Les joueurs devront chacun leur tour jouer. A chaque coup, on vérifie si il a gagné suivant l'une des 2 façons.

```
Fonction : jouerUnCoup (grille, joueurCourant,j1,j2) : booléen
D :  joueurCourant : joueur  {Le joueur Courant}
       grille matrice carrée de caractères de [0..18][0..18]
D/R : j1,j2 de type joueur {On met a jour les structures}
L :    lignesaisie, colonnesaisie : entier {Coordonnées du point qu'il va saisir}
       gagne : booléen
```

```
{init}
gagne :=FAUX;

  {Saisie du coup}
  saisie(grille,joueurCourant,lignesaisie,colonnesaisie)
  afficheGrille(grille);

  {On verifie si il a entoure son adversaire}
  testEntoure(grille,lignesaisie,colonnesaisie,&joueurCourant,j1,j2);

  {vérification si il a gagne suivant l'une des 2 conditions (nombre de prises ou pions alignes)}
  gagne=testGagne(grille,lignesaisie,colonnesaisie) ou joueurCourant.nbpionsadv==10

  // On renvoie le resultat
  Resultat := gagne;
Fin Fonction
```

Cette procédure permet de changer le joueur courant. Le joueur courant sera toujours celui qui doit jouer

```
Action : ChangerJoueur (grille, joueurCourant,j1,j2)
D :  j1,j2 de type joueur
       grille matrice carrée de caractères de [0..18][0..18]
D/R : joueurCourant : joueur  {Le joueur Courant}
L :   /
  Si (joueurCourant.num = j1.num) Alors
    joueurCourant=j2
  Sinon
    joueurCourant=j1
  Fin Si
Fin Action
```

Procédure principale :

Cette procédure lancera toutes les procédures du jeu

```
Action : pente
D :
D/R :
R :
L : grille matrice carrée de caractères de [0..18][0..18]
      j1,j2, joueurCourant de type joueur
      gagne : booléen
{init}
  gagne : =FAUX
  initGrille(grille);

  { Le premier aura le pion O et commencera la partie}
```

```
 j1.num=1;
 j1.pion='O';
 j2.num=2;
 j2.pion='X';
 creerJoueur(&j1);
 creerJoueur(&j2);

 afficheGrille(grille);

 { Le joueur j1 demarre la partie}
 joueurCourant=j1;

 {Logique du jeu : On s'arrête des que l'on a un gagnant ou si la grille est pleine }
 Tant que(!gagne et !grillePleine(grille)) faire
    { Le joueur courant joue un coup }
    gagne=jouerUnCoup(joueurCourant,j1,j2,grille);

    { Si le joueur n'a pas gagne, on change}
    Si (!gagne) Alors
        changerJoueur(grille,&joueurCourant,j1,j2);
    Fin Si
Fait

 { On affiche le résultat de la partie}
 Si (gagne) Alors
   Afficher(« Vous avez gagne joueur joueur.num »);
 Sinon
   Afficher("Match Nul : La grille est pleine et n'a pas de gagnants ");
 Fin Si
Fin Action
```

7. Compilation et utilisation du programme

La commande de compilation utilisée est sous LINUX : **gcc –o pente pente.c**

Pour lancer le programme il suffit de taper : **./pente**

Ce logiciel a été testé sous différentes plateformes : Linux version Debian et Knoppix et sous windows à l'aide du compilateur gcc de dev-C++

8. Listing du programme

```
/*
 * Cremetz Olivier
 * Mssiaidi Ali
 * Projet Algorithmique - Tp Jeu du Pente
 * A rendre pour le 26/11/2004
 */


#include <stdio.h>
#include <stdlib.h> // pour la fonction system

// TAILLE DE LA GRILLE
#define TAILLE 19

// NOMBRE DE PIONS ALIGNES POUR GAGNER
#define PIONS_ALIGNE 5
#define PIONS_ADVERSAIRE 10


/* CONSTANTES POUR DEFINIR UN BOOLEEN EN C */
#define VRAI 1
#define FAUX 0


/* Constantes pour definir les directions */
#define LIGNE 1
#define COLONNE 2
#define DIAG_MONTANTE 3
#define DIAG_DESCENDANTE 4



/*
  Structure permettant de définir un joueur
  il possède un numéro
  un prenom
  un pion lui sera affecté soit X ou O
  et nbpionsadv: le nombre de pions pris à son adversaire
*/
typedef
  struct{
    int num;
    char* prenom[20];
    char pion;
    int nbpionsadv;
   }joueur;




/****************************** INITIALISATION *********************************/


/*
  Action start
  Parametre en donnee ou Resultat ou donnee/Resultat : /
  Cette procedure permet d'afficher un ecran de presentation du jeu
  Cette procedure attend juste que l'utilisateur appuie sur une touche
*/

void start()
{
  printf("\n\t\t\t CREMETZ OLIVIER ET MSSIAIDI ALI \n");
  printf(" \n\t\t\t\t TP ALGORITHMIQUE\n");
  printf(" \n\t\t\t\t   JEU DU PENTE\n");

  printf("\n\n\n\t\t\t Appuyez sur une touche pour continuer...");
```

```
  getchar();
}


/*
  Action creerJoueur
  Parametre en donnee/Resultat: joueur j de type joueur
  Cette procedure permet au joueur d'entrer son prenom et d'initialiser
  le compteur de prises de pions a l'adversaire
*/
void creerJoueur(joueur *joueurCourant)
{
// On cree un joueur avec son nom  et on lui initialise son nombre de prises
  joueurCourant->nbpionsadv=0;
  printf("Entrer votre prenom joueur %d : ",joueurCourant->num);
  scanf("%s",&joueurCourant->prenom);
}



/*
  Action initGrille
  Parametre en donnee/resultat : grille matrice carree de caracteres de TAILLE
  Cette procedure initialise le tableau avec un espace
*/

void initGrille(char grille[TAILLE][TAILLE])
{
  int ligne,colonne;
  for (ligne=0;ligne<TAILLE;ligne++)
  {
    for (colonne=0;colonne<TAILLE;colonne++)
    {
      grille[ligne][colonne]=' ';
    }
  }
}


/*
  Action afficheGrille
  Parametre en donnee : grille matrice carree de caracteres de TAILLE
  Cette procedure permet d'afficher la grille du jeu
  avec les coups déja saisis
  Seuls les pions X et O sont affichés sinon on affiche un point
*/
void afficheGrille(char grille[TAILLE][TAILLE])
{
  int ligne,colonne;

  //On efface l'ecran
  system("clear");

  printf("**********************  JEU DU PENTE  **********************\n");
  printf("%5c",' ');
  for (colonne=0;colonne<TAILLE;colonne++)
  {
    printf("%3d",colonne+1);
  }
  printf("\n");

  for (ligne=0;ligne<TAILLE;ligne++)
  {
    printf("%3d  ",ligne+1);
    for (colonne=0;colonne<TAILLE;colonne++)
    {
```

```
        if(grille[ligne][colonne]=='X' || grille[ligne][colonne]=='O')
            printf("%3c",grille[ligne][colonne]);
        else
            printf("%3c",'.');
    }
    printf("\n");
}
}


/*
  Action saisie
  Parametre en donnee : joueurCourant de type Joueur
  Parametre en donnee resultat grille matrice carree de caracteres de TAILLE
  Parametre en resultat : lignesaisie et colonnesaisie de type entier
  Donne les coordonnees du coup joue
  Cette procedure permet de placer au joueur de placer son pion sur la grille du jeu
  Si un pion est deja a cet endroit l'ordinateur lui demandera de rejouer le coup
  Les coordonnees doivent bien sur entrer dans la grille
  On saisie d'abord la ligne puis la colonne
*/

void saisie(char grille[TAILLE][TAILLE],joueur joueurCourant,int *lignesaisie,int
*colonnesaisie)
{
int ligne,colonne;
int recommencer=VRAI;

while (recommencer){
printf("Joueur %d : entrez la position \nen ligne :",joueurCourant.num);
scanf("%d",&ligne);
printf("en colonne :");
scanf("%d",&colonne);

// Si les coordonnees sont en dehors du tableau ou si la place n'est pas libre, on
recommence la saisie
if(ligne>=1 && ligne<=TAILLE && colonne>=1 && colonne<=TAILLE && grille[ligne-1][colonne-
1]!='O' && grille[ligne-1][colonne-1]!='X')
    {
    recommencer=FAUX;
    }
    else{
    printf("Saisie incorrecte veuillez recommencer\n");
    recommencer=VRAI;
    }

}
//on affiche sur la grille la nouvelle position
grille[ligne-1][colonne-1]=joueurCourant.pion;

//on renvoie la position saisie
*lignesaisie=ligne;
*colonnesaisie=colonne;
}




/***************** SERIE DE TEST POUR GAGNER ***************************/


/*
  Action deplacement
  parametre en donnee : direction de type entier (voir les constantes)
  paramètres en resultat : deplacLigne,deplacColonne de type entier
  Cette procedure permet de renvoyer le point de départ pour tracer
  la droite
  en ligne : à gauche
  colonne : en haut
  diagonale montante : coin bas gauche
  diagonale descendante : coin haut gauche
*/
```

```
void deplacement(int direction, int *deplacLigne,int *deplacColonne)
{
if(direction==LIGNE)
{
// En ligne on part vers la gauche
*deplacLigne=0;
*deplacColonne=-1;
}

if(direction==COLONNE)
{
// En Colonne on part vers le haut
*deplacLigne=-1;
*deplacColonne=0;
}

if(direction==DIAG_MONTANTE)
{
// En Diagonale montante on part vers le bas gauche
*deplacLigne=1;
*deplacColonne=-1;
}

if(direction==DIAG_DESCENDANTE)
{
// En Diagonale descendante on part vers le haut gauche
*deplacLigne=-1;
*deplacColonne=-1;
}

}
```

```
/*
  Fonction testJeuGagne renvoie un entier
  Parametres en donnee:  grille :  matrice carree de caracteres de TAILLE
  lignesaisie, colonnesaisie : entiers representant le dernier coup joué.
  direction : Direction à teste (LIGNE COLONNE DIAG_MONTANTE DIAG_DESCENDANTE) voir les
constantes
  Cette fonction permet de tester si on a le nombre de pions alignes
  Pour des raisons de simplicite, on vérifie dans un sens (en ligne : à gauche,
  colonne : en haut, diagonale montante : coin bas gauche, diagonale descendante : coin haut
gauche)
  si on a les 5 pions alignes puis on calcul le nombre de pions dans l'autre sens
*/

int testJeuGagne(char grille[TAILLE][TAILLE],int lignesaisie,int colonnesaisie,int
direction)
{
{
int compteur=0;
int compteurtemporaire=0;
char pion_a_tester=grille[lignesaisie-1][colonnesaisie-1];

// on transpose le coup sur la matrice
int i=lignesaisie-1;
int j=colonnesaisie-1;

int deplacLigne,deplacColonne;
deplacement(direction, &deplacLigne,&deplacColonne);

//on deplace dans la premiere direction (jusqu'au premier pion appartenant au joueur)
// Quand on se déplace par la gauche on n'a pas besoin de teste si la colonne touche le cote
droit
while(i>=0 && j>=0 && i<TAILLE && grille[i+deplacLigne][j+deplacColonne]==pion_a_tester)
{
i=i+deplacLigne;
```

```
    j=j+deplacColonne;
    //On compte au cas ou on aurait tout dans le sens inverse
    compteurtemporaire=compteurtemporaire+1;
  }


  // Si on a deja le compte, on retourne directement le resultat on ne rentre pas dans la 2eme
boucle
  if(compteurtemporaire==PIONS_ALIGNE)
    return VRAI;

  //on va dans l'autre sens
  // Quand on se déplace par la droite on n'a pas besoin de teste si la colonne touche le cote
gauche
  while(compteur!=PIONS_ALIGNE && grille[i][j]==pion_a_tester && i>=0 && i<TAILLE &&
j<TAILLE)
  {
    compteur=compteur+1;
    i=i-deplacLigne;
    j=j-deplacColonne;
  }

  // Si on a le compte, on a gagne
  if(compteur==PIONS_ALIGNE)
    return VRAI;
  return FAUX;
}



/*
  Procedure testGagne
  Parametres en donnee:  grille :  matrice carree de caracteres de TAILLE
  lignesaisie, colonnesaisie : entiers representant le dernier coup joué par le joueur j.
  On lance la fonction testJeuGagne dans les 4 directions (voir les constantes LIGNE COLONNE
DIAG_MONTANTE et DIAG_DESCENDANTE)
*/
int testGagne(char grille[TAILLE][TAILLE],int lignesaisie,int colonnesaisie)
{
  int i;
  i=1;
  // On test les 4 directions
  while(i<=4 && !testJeuGagne(grille,lignesaisie,colonnesaisie,i))
  {
    i=i+1;
  }
  // Si la boucle s'est arretee avant de faire toutes les directions, on a gagne
  if(i<=4)
    return VRAI;

  return FAUX;
}



/********************************** ENTOURAGE D'UNE PAIRE DE PION ADVERSE ********/

/*
  Procedure supprimePion
  Parametres en donnee/resultat:
  grille :  matrice carree de caracteres de TAILLE
  coordV, coordH : entiers representant les coordonnees de la position a supprimer.
  On replace la case de la grille par un caractere espace
*/
void supprimePion(char grille[TAILLE][TAILLE],int coordV,int coordH)
{
  grille[coordV][coordH]=' ';
}
```

```
/*
  Procedure testEntoure
  Parametres en donnee:
  grille :  matrice carree de caracteres de TAILLE
  Cette grille possède tous les coups déja joués.
  lignesaisie, colonnesaisie : entiers representant le dernier coup joue par le joueur j.

  Parametre en donnee/resultat : joueurCourant,j1,j2 : de type défini joueur. La structure
permet de récuperer son pion, son nombre
  de pions pris a l'adversaire.

*/
void testEntoure(char grille[TAILLE][TAILLE],int lignesaisie,int colonnesaisie,joueur
*joueurCourant,joueur *j1,joueur *j2)
{
// On transpose les coordonnées representant le dernier coup joue par le joueur j dans la
grille
int i;
int j;
i=lignesaisie-1;
j=colonnesaisie-1;

// On recupere le pion du joueur courant
char pion=joueurCourant->pion;

// On recupere le pion de l'adversaire
char pionentoure;
if(pion=='O')
  pionentoure='X';
else
  pionentoure='O';

// coordonnees du point de depart
int deplacLigne,deplacColonne;
// sens pour le déplacement
int sens;

// indice de boucles
int m;
int n;

// on teste dans les 4 directions
for (m=1;m<=4;m++)
{
deplacement(m, &deplacLigne,&deplacColonne);

// On teste dans la direction des deux cotes
for (n=0;n<2;n++)
{
if (n==0)
  sens=1;
else
  sens=-1;

//On recupere le sens du deplacement
deplacement(m, &deplacLigne,&deplacColonne);

if(i+3*sens*deplacLigne >= 0 && i+3*sens*deplacLigne <= TAILLE &&
j+3*sens*deplacColonne>= 0 && j+3*sens*deplacColonne <= TAILLE &&
(grille[i+sens*deplacLigne*1][j+sens*deplacColonne*1]==pionentoure) &&
(grille[i+sens*deplacLigne*2][j+sens*deplacColonne*2]==pionentoure) &&
(grille[i+sens*deplacLigne*3][j+sens*deplacColonne*3]==pion))
{

// On incremente le nombre de pions du joueur Courant et celui du joueur
if (joueurCourant->num==j1->num){
  j1->nbpionsadv=j1->nbpionsadv+2;
  joueurCourant->nbpionsadv=joueurCourant->nbpionsadv+2;
```

```
        }
        else{
            j2->nbpionsadv=j2->nbpionsadv+2;
            joueurCourant->nbpionsadv=joueurCourant->nbpionsadv+2;
        }

        // On supprime la paire de pion que l'on a entoure
        supprimePion(grille,i+sens*deplacLigne*1,j+sens*deplacColonne*1);
        supprimePion(grille,i+sens*deplacLigne*2,j+sens*deplacColonne*2);
        afficheGrille(grille);
    }
  }
}
```

```
/*
  Fonction grillePleine : retourne un entier(booleen)
  Parametre en donnee : grille Matrice carree de caractere de TAILLE
  Cette procedure permet d'afficher de verifier si la grille est pleine ou pas
  On recherche Si une position ne contient pas un pion X ou O (Cela signifiera que la
  grille n'est pas pleine)
*/
int grillePleine(char grille[TAILLE][TAILLE])
{
  int i,j;

  for(i=0;i<TAILLE;i++)
  {
    for (j=0;j<TAILLE;j++)
    {
    // Si une position de la grille ne contient pas les pions X et O     on s'arrete
    if(grille[i][j]!='O' && grille[i][j]!='X'){
      return(FAUX);
    }

    }
  }
  return(VRAI);
}
```

```
/************************************  JOUER LE COUP ************************************/


/*
  Fonction jouerUnCoup : retourne un entier (booleen)
  Parametre en donnee: joueur joueurCourant : Le joueur Courant
  grille : Matrice carree de caractere de TAILLE
  Parametre en donnee/resultat : j1,j2 de type joueur (On met a jour les structures)
  Cette procedure permet au joueur courant d'entrer un coup
  et de verifier si il a gagne suivant l'une des 2 conditions (nombre de prises
  ou pions alignes)
*/
int jouerUnCoup(joueur joueurCourant,joueur *j1,joueur *j2,char grille[TAILLE][TAILLE])
{
  // Coordonnees du point qu'il va saisir
  int lignesaisie,colonnesaisie;

  int gagne=FAUX;

  printf("Joueur %d : %s [%c]\t\t Nombre de prises :
%d\n",joueurCourant.num,joueurCourant.prenom,joueurCourant.pion,joueurCourant.nbpionsadv);
  // Saisie du coup
  saisie(grille,joueurCourant,&lignesaisie,&colonnesaisie);
  afficheGrille(grille);
```

```
  // On verifie si il a entoure son adversaire
  testEntoure(grille,lignesaisie,colonnesaisie,&joueurCourant,j1,j2);

  //verification si il a gagne suivant l'une des 2 conditions (nombre de prises ou pions
alignes)
  gagne=testGagne(grille,lignesaisie,colonnesaisie) ||
joueurCourant.nbpionsadv==PIONS_ADVERSAIRE;

  // On renvoie le resultat
  return gagne;
}
```

```
/*
  Action changerJoueur
  Parametre en donnee: joueur j1,j2 : Les joueur de la partie
  grille : Matrice carree de caractere de TAILLE
  Parametre en donnee/Resultat : joueurCourant de type joueur
  Cette procedure permet de changer le joueur courant
*/
void changerJoueur(char grille[TAILLE][TAILLE],joueur *joueurCourant,joueur j1, joueur j2)
{
  if(joueurCourant->num==j1.num)
    *joueurCourant=j2;
  else
    *joueurCourant=j1;
}
```

```
/*
  Fonction main renvoie un entier (0 pour annoncer que le programme s'est bien deroule
  Dans cette fonction, on affiche l'ecran de demarrage, on initialise la grille
  on cree les joueurs puis on fait joueur les joueurs jusqu'a temps qu'ils aient gagnes ou
  que la grile soit pleine
*/

int main()
{
  char grille[TAILLE][TAILLE];
  int gagne=FAUX;

  start();
  initGrille(grille);

  // On cree 2 joueurs
  // Le premier aura le pion O et commencera la partie
  joueur j1;
  joueur j2;
  j1.num=1;
  j1.pion='O';
  j2.num=2;
  j2.pion='X';
  creerJoueur(&j1);
  creerJoueur(&j2);

  afficheGrille(grille);

  // On cree un joueur courant
  joueur joueurCourant;
  // Le joueur j1 demarre la partie
  joueurCourant=j1;
```

```
// Logique du jeu
// On s'arrete des que l'on a un gagnant ou si la grille est pleine
while(!gagne && !grillePleine(grille))
{
// Le joueur courant joue un coup
gagne=jouerUnCoup(joueurCourant,&j1,&j2,grille);

// Si le joueur n'a pas gagne, on change
if(!gagne)
    changerJoueur(grille,&joueurCourant,j1,j2);
}

// On affiche le resultat de la partie
system("clear");
if(gagne){
    printf("\n\nFelicitation %s, joueur %d: Vous avez gagne avec les pions
%c\n",joueurCourant.prenom,joueurCourant.num,joueurCourant.pion);
}
else
    printf("\n\nMatch Nul : La grille est pleine et n'a pas de gagnants\n");

return 0;
}
```

9. Conclusion

La réalisation de ce programme nécessite de bonnes bases en algorithmique. Ce projet nous a donc permis de prendre conscience de l'importance de l'analyse et de la répartition des tâches dans le cas de la réalisation d'un projet informatique. Nous avons essayé de faire de ce programme une synthèse de toutes les notions vues en cours d'algorithmique du 1er semestre.
Le programme réalisé pourra être étendu vers d'autres cours comme la gestion du jeu en réseau, ou un système de meilleurs scores en base de données.