TP – Tableaux Statiques
ING1 – Programmation C
Génie Mathématiques

EISTI

Préambule

L'ensemble du TP formera un seul programme. Les différentes fonctionnalités traitées devront être accessible via un menu. Je vous conseille de bien séparer chacune des fonctions, ainsi que celle du menu, de la saisie ... Bien découper votre code vous facilitera la lisibilité du code, et vous évitera un certain nombre d'erreurs.

1 Vecteur

Vous travaillerez sur des tableaux d'entiers de taille fixe. La taille sera donnée par une constance symbolique. Pour rappel, une constante symbolique se définit par :

```
#define N 20
```

(1) **Inversion**
Écrire une procédure qui inverse les éléments d'un tableau (sans passer par un tableau temporaire) : le premier devient le dernier, le deuxième devient l'avant dernier, etc. □

(2) **Somme**
Écrire une procédure qui prend en paramètre trois vecteurs : les deux à additionner, et le résultat de l'addition. □

(3) **Produit scalaire**
Écrire la fonction qui permet de calculer le produit scalaire de deux vecteurs passés en paramètre. □

(4) **Produit vectoriel**
Écrire une **procédure** qui, à partir de deux vecteurs passés en paramètres, renvoie le résultat du produit vectoriel.
Attention, pour le moment, vous ne savez pas retourner un tableau, donc vous devrez utiliser une procédure avec trois paramètres. □

2 Tableaux 2D – Tic-Tac-Toe

Le Tic-tac-toe se joue sur une grille carrée de 3×3 cases. Deux joueurs s'affrontent. Ils doivent remplir chacun à leur tour une case de la grille avec le symbole qui leur est attribué : O ou X. Le gagnant est celui qui arrive à aligner trois symboles identiques, horizontalement, verticalement ou en diagonale. Pour mettre en place le jeu, nous allons découper en plusieurs fonctions les différentes actions à faire [1].

La structure utilisée pour modéliser le plateau sera un tableau d'entiers à deux dimensions. Puisque la taille est toujours de 3, on utilisera un tableau statique. Les valeurs possibles du plateau seront : -1 si personne n'a encore joué à cette case, 1 ou 2 si le joueur respectivement 1 ou 2 a joué à cette case.

Initialisation

Au tout début du jeu, les cases du plateau seront initialisées à -1, pour signifier qu'aucun joueur n'a joué. Implémenter la méthode d'initialisation :

```
void init (int ttint_plateau[N][N]);
```

Affichage

À chaque tour de jeu, le plateau sera affiché afin de visualiser l'avancement de la partie. Implémenter la méthode `affichage` qui écrit 'X' lorsque le joueur 1 a joué sur la case, 'O' s'il s'agit du joueur 2, ou alors ' ' s'il n'y a personne.

```
void affichage (int ttint_plateau[N][N]);
```

Jouer

Les joueurs jouent tour à tour. Écrire une méthode `jouer` qui permet à un joueur donné de jouer sur une case spécifique. Si la case est déjà occupée par l'autre joueur, afficher un message d'erreur. La méthode retournera 1 si tout s'est bien passé, et 0 si une erreur est survenue.

```
int jouer (int ttint_plateau[N][N], int int_joueur, int int_x, int int_y);
```

1. Vous avez tout à fait la possibilité de créer d'autres méthodes en fonction de vos besoins.

A gagné

À la fin de chaque tour de jeu, il faut vérifier si l'un des deux joueurs a gagné. Implémenter la fonction `aGagne` , qui retourne soit le numéro du joueur gagnant, soit 0 si c'est match nul, soit -1 si la partie n'est pas terminée.

```
int aGagne (int ttint_plateau[N][N]);
```

Tour de jeu

Écrire la méthode `tourDeJeu` qui permet de faire jouer les joueurs chacun leur tour, jusqu'à la fin de la partie. Lorsque la partie est finie, la grille complète devra être affichée, ainsi que le vainqueur.

Lors d'une erreur de saisie de la part de l'utilisateur, le programme devra demander une nouvelle saisie, jusqu'à ce que l'utilisateur saisisse quelque chose de cohérent.

```
void tourDeJeu (int ttint_plateau[N][N]);
```