LIFAP1 – TP7 : Tableaux à 2 dimensions

Objectifs : Manipulation des tableaux à 2 dimensions
Utilisation de tailles variables dans les tableaux

1. Tableaux 2D : la base
   a. Soit un tableau d'entiers à deux dimensions de taille maximum `TAILLE_LIGNE` et `TAILLE_COLONNE`. Écrivez une procédure qui, à partir du nombre de lignes et de colonnes données par l'utilisateur remplit les `tailleL * tailleC` cellules de ce tableau.
   b. Écrivez une procédure qui affiche (proprement) le contenu du tableau précédent.

2. Le morpion : des caractères dans des tableaux !!!

Dans cette partie nous allons programmer le jeu du morpion. Pour cela, vous avez besoin d'une grille 3*3 et de 2 joueurs ayant des pions différents (les croix et les ronds).

X	X	O
O	X	X
X	O	O

Grille sans gagnant

X	O	X
O	X	O
O	X	X

Grille avec gagnant

A tour de rôle, chaque joueur positionne un de ses pions sur la grille. Le jeu se finit quand un joueur a réalisé une ligne, une colonne ou une diagonale avec ses pions (c'est le gagnant) ou quand la grille est pleine (pas de gagnant).
La grille est représentée par un tableau à 2 dimensions de caractères dont chaque case contiendra soit '_', soit 'O', soit 'X'. Pour réaliser l'implémentation de ce jeu, écrivez les sous-programmes suivants.

   c. Initialisation de la grille du morpion à vide (caractère '_')
      `void initialiseGrille(char grille[3][3])`
   d. Affichage de la grille du morpion : _ indique case vide, O pion joueur 1 et X pion joueur 2 :
      `void afficheGrille(char grille[3][3])`
   e. Saisie des coordonnées du nouveau pion à mettre sur la grille. Si les coordonnées sont en dehors de la grille ou si la case possède déjà un pion, la saisie est refusée, un message d'erreur est affiché, et le joueur doit rejouer. Dans le cas où les coordonnées sont correctes, placer le pion sur la grille à cet emplacement.
      `void metUnPionSurLaGrille(char grille[3][3], char &joueur)`
   f. Teste si l'un des joueurs a gagné (ligne, colonne ou diagonale remplie de pions semblables). Dans ce cas, affiche un message pour indiquer le joueur qui a gagné. S'il n'y a pas de gagnant, teste que la grille n'est pas pleine. Si elle est pleine, affiche un message indiquant qu'aucun des joueurs n'a gagné. Retourne TRUE si la grille est pleine ou si un joueur a gagné, FALSE sinon.
      `bool testeFinJeu(char grille[3][3], char joueur)`
   g. Écrivez ensuite le programme principal permettant de dérouler la partie. En voici son algorithme :

```
Algorithme principal :
   Initialisation de la grille à vide
   Tant que (pas de gagnant ou pas grille pleine)
    Afficher grille
    Mettre un pion sur la grille
   Fin Tant que
```