Programmation C
TD/TP7 : Révisions

Pré-requis : tout le cours de Programmation Bas Niveau.

Exercice 1

Définir une structure puissance4 et son raccourci P4. Elle servira à encoder les informations de la partie : l'état actuel, le numéro du coup, le nom de chacun des joueurs. Cette structura contiendra donc :

— deux int pour coder la largeur et la hauteur de la grille ;
— deux char * pour le nom de chacun des joueurs ;
— un int ** (que nous appellerons grille) pour coder la grille ;
— un int pour représenter le numéro du coup en cours.

Exercice 2

Coder une fonction **P4 initialisePartie (int largeur, int hauteur, char* joueur1, char* joueur2)** qui renvoie une partie initialisée. Initialiser une partie signifie : allouer de la mémoire au tableau de pointeurs grille. Pour ce faire, il faut penser d'abord à allouer de la mémoire pour stocker chaque ligne, puis à allouer de la mémoire à chaque ligne (grille est un pointeur sur un pointeur de type int, il peut donc encoder un tableau à deux dimensions). La fonction devra également initialiser toutes les cases de la grille à 0, la hauteur et la largeur d'après les paramètres, et copier les chaînes de caractères **joueur1** et **joueur2**. Pour rappel, la fonction doit pour cela allouer aux chaînes de la structure **P4** la taille nécessaire pour stocker **joueur1** et **joueur2**. Le numéro du coup doit être fixé à 1.

Exercice 3

Coder une fonction **int poserPiece (P4 * partie)** qui renvoie -1 si la colonne est pleine. Si la colonne n'est pas pleine, alors la fonction place une pièce dans la première case disponible (la pièce doit être du numéro du joueur (les joueurs jouent en alternance, il est donc possible de connaître le numéro du joueur d'après le numéro du coup), incrémente le numéro du coup et renvoie le numéro de la case à laquelle la pièce a été posée.

Exercice 4

Coder une fonction **void afficheGrille (P4* partie)** qui affiche la grille en cours. Les colonnes doivent être séparées par le caractère — (altGr + 6) et afficher O pour une pièce du joueur 1, et X pour une pièce du joueur 2. Tester les fonctions des trois premiers exercices (initialisation, pose succesive de pièces, affichage).

Exercice 5

Coder une fonction **int verifieGagne (P4* partie)** qui renvoie 0 si personne n'a gagné, le numéro du joueur qui a gagné sinon. Rappel : pour gagner au puissance 4, il faut aligner 4 pièces horizontalement, verticalement ou en diagonale.

Exercice 6

Coder une fonction **void affichePartie (P4* partie)** qui affiche l'état d'une partie en cours : elle doit afficher le message « {Nom_du_joueur} : à toi de jouer ! Choisis une colonne où poser ta pièce ! » ainsi que la grille (utilisez la fonction précédemment codée).

Exercice 7

Ecrire un programme qui fait appel à l'affichage, et aux fonctions précédentes afin de demander aux joueurs, en boucle, une colonne où poser leur pièce jusqu'à ce que la grille soit pleine (cela peut se vérifier grâce au nombre de coups joués) ou qu'un joueur ait gagné. Le puissance 4 standard fait 7 cases de large pour 6 de haut. Attention à bien prévoir des tableaux assez grands pour pouvoir y stocker les noms des joueurs que l'on va passer à la fonction d'initialisation.

Exercice 8

Coder une fonction **int sauvegarder (P4* partie, char* fichier)** qui sauvegarde la partie dans le fichier de nom **fichier**. Attention, il y a plusieurs choses différentes à sauvegarder : les dimensions de la grille, les noms des joueurs, le nombre de coups et la grille elle-même. La fonction devra renvoyer **-1** si la sauvegarde a échoué, **1** sinon. Il faudra faire plusieurs appels différents à fwrite pour écrire chacun des composants, et non un appel à fwrite pour écrire une variable de type P4, qui sauvegarderait uniquement des pointeurs et non le contenu des variables pointées. De la même manière, il ne faudra pas écrire des char* pour les noms, mais chaque caractère composant le nom, sans oublier d'écrire un caractère qui permette d'identifier clairement la fin de la chaîne (par exemple '\0').

Exercice 9

Coder la fonction inverse à la fonction précédente : **int charger (P4* partie, char* fichier)** qui charge la partie sauvegardée dans le fichier de nom **fichier** grâce à la fonction précédente, et la place dans **partie**.

Exercice 10

Modifier le programme afin de sauvegarder la partie à chaque coup joué. Au début du programme, proposer le chargement de la dernière partie ou l'initialisation d'une nouvelle partie.