

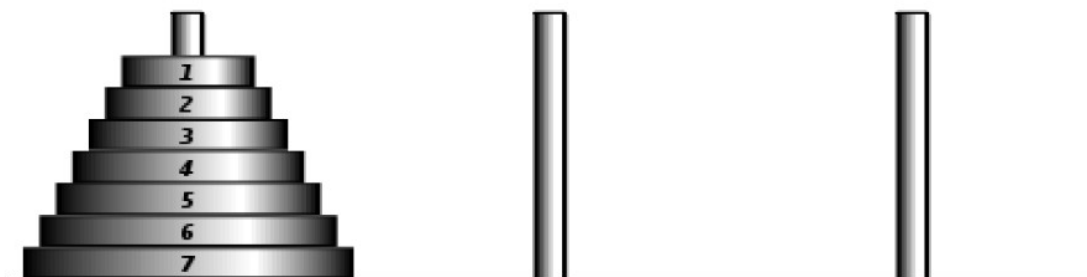
Travaux Pratiques 2
Structures de données élémentaires
Les tours de Hanoï

L'objectif de ce TP est d'implémenter le fonctionnement des célèbres tours de Hanoï. Ces dernières, aussi appelées Tours de Brahma ou Puzzle de la fin du monde, ont été inventées par le mathématicien Français Edouard Lucas en 1883 qui s'est inspiré d'une légende :

« Dans un temple Hindou, les prêtres reçurent trois poteaux, le premier supportant une pile de 64 disques d'or, chacun un peu plus petit que celui d'en-dessous. Ils devaient transférer les 64 disques d'un poteau à l'autre, avec une seule règle : un disque ne pouvait pas être placé sur un autre disque plus petit. Toujours d'après la légende, la fin du monde n'arriverait qu'après qu'ils aient fini leur travail. »

Comme nous n'avons pas particulièrement l'intention de détruire le monde, nous nous contenterons de n disques avec $n < 8$. Récapitulons les règles :

- le jeu est constitué de trois tours ;
- n disques de taille toutes différentes sont empilées sur l'une des tours, par tailles décroissantes avec le plus grand en bas ;
- chaque disque est forcément sur une des trois tours ;
- un disque ne peut être posé sur un disque plus petit. Par contre, il peut être posé sur un disque plus grand ;
- un seul disque peut être déplacé à la fois.



L'objectif de ce TP est donc d'implémenter le jeu des tours de Hanoï en suivant différentes étapes. Premièrement, il faut créer une structure de pile.

1) Créez les fichiers **pile.c** et **pile.h** et implémentez-y une structure de pile d'entiers par tableau ainsi que les procédures de pile suivantes :

- **créerPile** : crée une pile vide de capacité maximale fixe en allouant la mémoire nécessaire ;
- **détruirePile** : détruit une pile en libérant ses ressources mémoire ;
- **capacitePile** : retourne la capacité maximale d'une pile (ne pas confondre avec la taille de la pile) ;
- **pileVide** : retourne vrai si la pile est vide, faux sinon ;
- **pilePleine** : retourne vrai si la pile est pleine, faux sinon ;
- **empiler** : ajoute un élément au sommet de la pile s'il reste de la place ;
- **dépiler** : enlève un élément du sommet de la pile s'il en existe un et retourne sa valeur ;
- **sommet** : retourne la valeur de l'élément du sommet de la pile sans modifier cette dernière.

Notez que pour une implémentation propre qui respecte les principes de la structure de pile, les fichiers **pile.c** et **pile.h** ne devraient pas contenir d'autres fonctions que celles demandées ci-haut. Par conséquent, tout le reste du code que vous développerez dans ce TP doit être à l'extérieur de ces fichiers et vous ne devez pas utiliser vos piles autrement que par ces fonctions (pas d'accès direct à la structure de pile à partir de l'extérieur).

2) Créez les fichiers **outilsPile.c** et **outilsPile.h** qui contiendront diverses fonctions utilitaires de manipulation des piles, et implémentez-y une fonction **afficherPile** qui affiche à l'écran les informations relatives à votre pile.

3) Créez un programme de test permettant de vérifier le fonctionnement de votre implémentation en demandant à l'utilisateur de saisir des valeurs entières, en les empilant successivement et en affichant la pile ainsi créée. Essayez d'empiler sur une pile pleine, de dépiler sur une pile vide, etc...

Maintenant que nous avons une structure de pile fonctionnelle, nous pouvons implémenter les tours de Hanoï.

3) Créez les fichiers **hanoi.h** et **hanoi.c** et implémentez-y une structure adéquate pour les tours de Hanoï ainsi que les fonctions suivantes :

- **creerHanoi** : crée et initialise un jeu des tours de Hanoï avec un nombre de disques spécifique, ces derniers étant empilés sur la première des trois tours ;
- **afficherHanoi** : affiche l'état actuel du jeu des tours de Hanoï avec les disques sur chaque tour. Vous pouvez créer votre propre affichage ou bien utiliser celui proposé par le professeur (**afficherHanoi.txt**, téléchargeable à partir du site web du cours : <http://cosy.univ-reims.fr/~pdelisle/enseignement.php>). Pour ce faire, vous devez implémenter les fonctions **copierPile** (copie une pile dans une autre pile) et **taillePile** (retourne la taille de la pile, c'est-à-dire le nombre d'éléments effectivement présents dans la pile, à ne pas confondre avec la capacité de la pile) dans votre fichier outilsPile.c en respectant le format des structures ou en les adaptant à votre code ;
- **deplacerDisque (int tourDepart, int tourArrivee)** : déplace un disque de la tour de numéro tourDepart vers la tour de numéro tourArrivee. La fonction ne doit effectuer le déplacement que si celui-ci est possible, donc s'il n'enfreint pas les règles définies précédemment ;
- **gagne** : vérifie si le jeu est gagné, donc si tous les disques sont empilés sur la deuxième ou la troisième tour.

4) Dans votre programme de test, écrivez un code permettant de jouer une partie du jeu des tours de Hanoï en entrant les déplacements au clavier et testez-le.

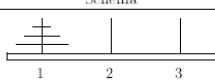
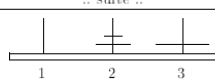
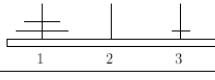
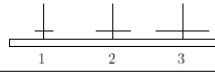
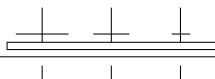
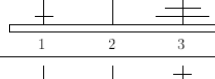
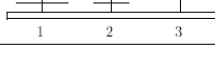
Questions supplémentaires pour les rapides et les motivés

- Écrivez un code permettant de jouer au jeu des tours de Hanoï de manière automatique en utilisant la méthode du chiffre d'un coup (la description de la méthode est donnée ci-bas).
- Écrivez des codes implémentant les files et les listes chaînées ainsi que leurs principales opérations (insertion, suppression, récupération d'un élément, affichage, etc...).

Annexe - La méthode du chiffre d'un coup

La méthode du chiffre d'un coup a été proposée en 1982 par Ivan Lavallée. Elle utilise le fait qu'il existe six déplacements possibles, ou coups, auxquels on associe un chiffre qui correspond à la somme des numéros des tours du coup (voir le tableau ci-bas, à gauche). Selon la configuration, il existe une seule manière de faire un déplacement, c'est-à-dire un seul coup possible associé à un chiffre. Ainsi, si on peut déplacer $1 \rightarrow 2$, alors $2 \rightarrow 1$ est impossible. La méthode du chiffre d'un coup impose que la suite des chiffres associés aux déplacements pour arriver à la solution est périodique et prend les valeurs 4-3-5. Un exemple avec 3 disques est donné dans la figure ci-bas, à droite.

Coup	Chiffre associé
$1 \rightarrow 2$	3
$2 \rightarrow 1$	3
$1 \rightarrow 3$	4
$3 \rightarrow 1$	4
$2 \rightarrow 3$	5
$3 \rightarrow 2$	5

Chiffre	Coup	Schéma	.. suite suite suite ..
			4	$1 \rightarrow 3$	
4	$1 \rightarrow 3$		3	$2 \rightarrow 1$	
3	$1 \rightarrow 2$		5	$2 \rightarrow 3$	
5	$3 \rightarrow 2$		4	$1 \rightarrow 3$	