CRO TP1 :

Les Tours de Hanoï

(1 séance sur machine)

1 But du TP

Le but de ce TP est de vous familiariser avec la manipulation des pointeurs dans le langage C. Ce TP introduit un exemple de *gestion dynamique* de la mémoire. La *décomposition modulaire* d'un projet logiciel est aussi illustrée ici, ainsi que l'écriture d'un Makefile complet.

2 Description

Il s'agit de programmer la résolution du problème des tours de Hanoï pour des tours de taille N (N n'étant pas forcément connu à la compilation). Les tours de Hanoï seront modélisées grâce à une structure de pile. La décomposition de programme en modules ainsi que les prototypes des fonctions manipulant les piles vous sont proposées en téléchargement (voir section 3).
On rappelle le principe des tours de Hanoï déjà vu cours ALG. Les tours de Hanoï sont 3 pics sur lesquels sont insérées N rondelles de tailles croissantes. Au départ les rondelles sont sur le pic de gauche. Le but est de toutes les déplacer sur le pic de droite. Par exemple, Les positions des rondelles au départ et à la fin, pour des tours de hanoï de taille 4 sont illustrées en figure 1.

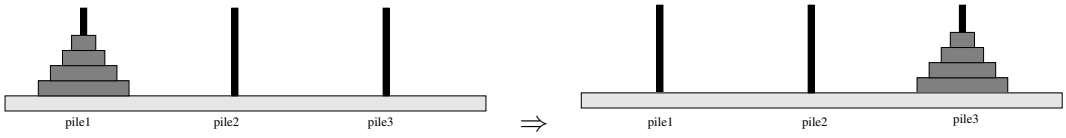

FIGURE 1 – résolution du problème des tours de Hanoï de taille 4

Une étape correspond au déplacement d'une rondelle, sachant que :

— On ne peut déplacer que les rondelles situées au sommet des tas.

— On ne peut poser une rondelle que sur une rondelle de plus gros diamètre (ou sur le socle).

On modélisera le problème en utilisant trois piles d'entiers pour les trois pics. Une rondelle est donc représentée par un entier (correspondant à la largeur de la rondelle). Le résultat du TP sera une fonction `hanoi` permettant de résoudre le problème de Hanoï en utilisant uniquement la fonction `déplacer` qui déplace une rondelle d'un pic à un autre.
On rappelle qu'une pile est une structure dans laquelle les ajouts et les retraits ont lieu au sommet de pile : l'élément dépilé est le dernier à avoir été empilé. En C, une pile est implémentée par une liste chaînée pour laquelle l'insertion et le retrait d'élément aura lieu *en tête de liste*. Pour ce TP, on utilisera le type suivant pour une pile :

```
/* Element de pile */
struct model_elem {
  int elem ;
  struct model_elem* suivant;
};
typedef struct model_elem ELEMPILE;

/* Pile */
typedef ELEMPILE *PILE;
```

3 A votre disposition

Vous ferez ce TP sous Linux en C. Le debogueur `ddd` est éventuellement utilisable. Vous pouvez télécharger une archive (fichier `TP1-init.tar`) sur Moodle. Ce fichier et les fichiers `pile_type.h`, `pile.h` et `hanoi.h` proposant une décomposition modulaire qui correspond à l'architecture logicielle illustrée sur la figure 1. Le travail du TP consistera essentiellement en l'écriture du fichier `pile.c` qui contiendra le code des fonctions dont le prototype est décrit dans le fichier `pile.h`, et l'écriture du fichier `hanoi.c`. Le fichier `main.c` présent dans l'archive sera utilisé pour tester les fonctions de manipulation de pile **au fur et à mesure de leur écriture**.
Pour extraire l'archive on peut exécuter la commande suivante : `tar -xvf TP1-init.tar`
Cela créé un repertoire `code`, dans lequel vous pouvez travailler.

4 Travail à faire

1. Récupérez l'archive sur Moodle et créer le Makefile (pour cela créer les fichiers `pile.c` et `hanoi.c`). **faites valider par l'enseignant**. Si vous n'avez pas fait de Makefile, vous ne devez pas continuer le TP.
2. Programmez les fonctions de manipulation d'une pile dans le fichier `pile.c` :
   (a) D'abord regarder le fichier `pile.h` pour comprendre ce que doit faire chaque fonction. Programmez la fonction `error1` et vérifiez qu'elle fonctionne avec votre `main`.
   (b) Commencez par la fonction `afficherPile` sans laquelle on ne peux pas vérifier que nos piles sont correctes.
   (c) Pogrammez ensuite la fonction `Empiler`, vérifiez son fonctionnement.
   (d) Enfin la fonction `Depiler`, attention vous remarquerez que la fonction `Depiler` prend en argument un pointeur sur une pile (donc un pointeur sur un pointeur sur un élément). La raison en est, bien évidement, qu'une fonction C ne peux pas modifier un argument. Cette fonction renvoyant déjà l'éléménet dépilé, elle ne peut en même temps renvoyer la pile modifiée. Ce mécanisme est utilisé **systématiquement dans le langage C** : lorsqu'une fonction modifie un de ses argument, on passe en argument un pointeur sur l'argument plutot que l'argument lui-même. Pour bien comprendre cela, on rappelle le principe des passages d'arguments par valeur en C, si vous aves encore des doutes avec le `PILE* pile`, demandez à un enseignant. Enfin, n'oubliez pas, lorsque vous dépilez, de libérer la mémoire avec la fonction `free`

passage de paramètre en C

Le mécanisme de passage de paramètre en C (on dit : passage de paramètre par valeur) doit être bien compris :

Passage de paramètre par valeur

- Ce qui est transmis à la fonction est *une copie de la valeur de l'argument*
- dans `int factorielle(int n)`, `n` est le *paramètre formel* de la fonction. Il peut être utilisé dans le corps de la fonction comme une variable locale.
- dans `x=factorielle(10);`, `10` est le *paramètre effectif* utilisé lors de cet appel.
- En C, tout se passe lors de cette appel comme si on exécutait le corps de la fonction *avec la case mémoire pour* `n` *contenant une copie de la case mémoire contenant 10*. On dit que les paramètres sont passés *par valeur*.
- Lorsque l'appel est terminé, la case mémoire de `n` disparait.
- Donc : on peut modifier la valeur du paramètre formel `n` dans le corps de la fonction mais cela ne modifiera pas la valeur du paramètre effectif (10).

Passage de paramètre par référence

- La seule manière, pour une fonction C, de modifier son argument et de travailler sur *un pointeur sur l'argument*. On parle alors de *passage par référence* (une *référence* est un autre nom pour un pointeur). C'est un abus de langage, le passage est toujours par valeur, mais on passe une référence sur l'objet.
- Ce point particulier (passage par valeur ou par référence) est un des point qui différencie les différents langages de programmation, il est important de comprendre ce que cela implique, ainsi que de comprendre comment cela est implémenté (juste par un pointeur... ou par une copie).

3. Programmer la résolution du problème des tours de Hanoï pour des tours de taille N, pour cela on utilisera les prototypes de fonctions donnés dans le fichier `hanoi.h`. Testez votre programme, mesurez les performances pour des valeurs croissantes de N.
4. Lorsque le programme sera réalisé, écrivez une fonction d'affichage de pile dédiée à ce TP qui affiche convivialement les tours à l'écran.