

## R1.01 : Initiation au développement (partie 2)

### Feuille TD n° 5

#### Utilisation de Piles

#### Objectifs :

1.- Apprendre les manipulations de base d'un Type abstrait de Données Pile

#### Révisions de cours

##### 1. Afficher le contenu d'une pile

On souhaite écrire un sous-programme, nommé **afficher**, qui affiche à l'écran le contenu d'une pile **pile** passée en paramètre (par exemple, un pile d'entiers).

#### TRAVAIL A FAIRE

1. Écrire en C++ la *déclaration* de ce sous-programme.
2. Définir la stratégie de cet algorithme.
  - (a) Modèle(s) d'algorithme(s) vu(s) en cours applicable(s) pour atteindre le but visé. *Justifier*.
  - (b) Action(s) à répéter
  - (c) La/les conditions de fin de répétition
3. Justifier les choix suivants : procédure/fonction et type de passage de paramètre choisi.
4. Écrire l'algorithme correspondant, accompagné d'un *dictionnaire succinct* des variables/constantes utilisées : nom, type, signification.

#### ...Un classique, en version itérative

##### 2. Résolution du problème des tours de Hanoï

Les tours de Hanoï<sup>1</sup> ont été inventées par un mathématicien français, Edouard Lucas en 1883. Sous le titre « Les brahmes tombent », Lucas relate que « *N. Claus de Siam a vu, dans ses voyages pour la publication des écrits de l'illustre Fer-Fer-Tam-Tam, dans le grand temple de Bénarès, au-dessous du dôme qui marque le centre du monde, trois aiguilles de diamant, plantées dans une dalle d'airain, hautes d'une coudée et grosses comme le corps d'une abeille. Sur une de ces aiguilles, Dieu enfila au commencement des siècles, 64 disques d'or pur, le plus large reposant sur l'airain, et les autres, de plus en plus étroits, superposés jusqu'au sommet. C'est la tour sacrée du Brahmâ. Nuit et jour, les prêtres se succèdent sur les marches de l'autel, occupés à transporter la tour de la première aiguille sur la troisième, sans s'écarter des règles fixes que nous venons d'indiquer, et qui ont été imposées par Brahma. Quand tout sera fini, la tour et les brahmes tomberont, et ce sera la fin des mondes !*<sup>2</sup> ».

Le jeu basé sur cette légende a pour but de bouger, en un minimum de coups, tous les disques de la tour de départ vers une nouvelle tour (tour d'arrivée) en se servant d'une tour intermédiaire, en respectant les règles suivantes :

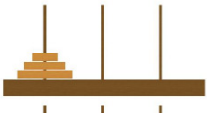

- On ne peut déplacer qu'un disque à la fois,
- On ne peut pas déplacer un disque qui se trouve sous un autre disque
- Un disque ne peut pas être déplacé sur un disque plus petit.

Nous allons résoudre le casse-tête pour un nombre N de disques de tailles différentes. La taille de chaque disque sera représentée par une valeur entière (de 1 à N).

<sup>1</sup> Aussi appelées Tours de Brahma ou Puzzle de la fin du monde

<sup>2</sup> [http://fr.wikipedia.org/wiki/Tours\\_de\\_Hanoï](http://fr.wikipedia.org/wiki/Tours_de_Hanoï)

Les tours seront numérotées 1 à 3, la tour de départ étant la tour n°1 et la tour d'arrivée la tour n°3.

<p>Configuration initiale :</p>  <p>Configuration finale :</p> 	<p>On souhaite écrire un programme qui saisit la valeur de N (taille du plus grand disque), et qui affiche ensuite à l'écran les différentes étapes (états de chaque tour) pour résoudre le problème avec N disques.</p>
---	--

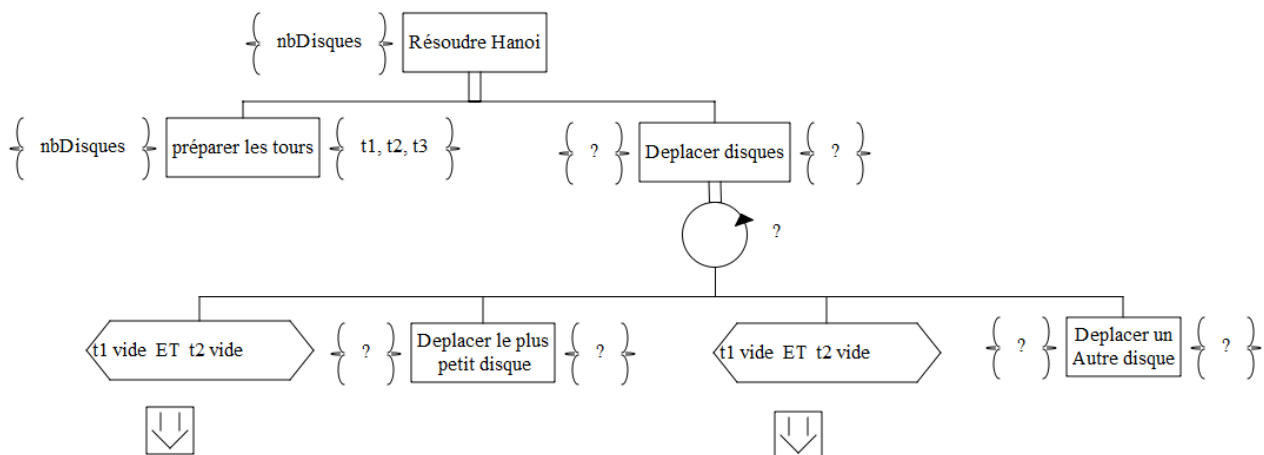
## Analyse du problème

En observant le jeu de plus près, on se rend compte que, quelle que soit la position des disques, il n'y a que 2 familles de déplacements possibles sur ces 3 tours :

- Le plus petit disque peut toujours se déplacer sur les 2 autres tours
- Si un disque différent du plus petit peut se déplacer, alors il ne peut se déplacer que sur 1 seule tour : celle où n'est pas le petit disque.

(justification : cf. Wikipédia)

Un algorithme itératif de résolution du problème serait alors



Avec :

Déplacement du plus petit disque : on choisira TOUJOURS le sens t1 -> t2 -> t3 -> t1

Déplacement de l'autre disque : on le déplacera où l'on peut (vu qu'il n'y a pas le choix).

Les copies d'écran ci-dessous montrent 2 exécutions de l'algorithme, avec 3 disques, puis 4 disques. Le programmeur se a ajouté 3 actions d'affichage des tours afin de visualiser l'évolution de leur contenu.

### Exemple d'exécution avec nbDisques=3 :

```

Entrer le nbre de disques de la tour de Hanoi (entre 3 et 9) : 3
t1= { 1. 2. 3. } t2= { } t3= { }

-----

t1= { 2. 3. } t2= { 1. } t3= { }
t1= { 3. } t2= { 1. } t3= { 2. }

t1= { 3. } t2= { } t3= { 1. 2. }
t1= { } t2= { 3. } t3= { 1. 2. }

t1= { 1. } t2= { 3. } t3= { 2. }
t1= { 1. } t2= { 2. 3. } t3= { }

t1= { } t2= { 1. 2. 3. } t3= { }
t1= { } t2= { 1. 2. 3. } t3= { }

t1= { } t2= { 2. 3. } t3= { 1. }
t1= { 2. } t2= { 3. } t3= { 1. }

t1= { 1. 2. } t2= { 3. } t3= { }
t1= { 1. 2. } t2= { } t3= { 3. }

t1= { 2. } t2= { 1. } t3= { 3. }
t1= { } t2= { 1. } t3= { 2. 3. }

t1= { } t2= { } t3= { 1. 2. 3. }

```

### Exemple d'exécution avec nbDisques=4 :

```

Entrer le nbre de disques de la tour de Hanoi (entre 3 et 9) : 4
t1= { 1. 2. 3. 4. } t2= { } t3= { }

-----

t1= { 2. 3. 4. } t2= { 1. } t3= { }
t1= { 3. 4. } t2= { 1. } t3= { 2. }

t1= { 3. 4. } t2= { } t3= { 1. 2. }
t1= { 4. } t2= { 3. } t3= { 1. 2. }

t1= { 1. 4. } t2= { 3. } t3= { 2. }
t1= { 1. 4. } t2= { 2. 3. } t3= { }

t1= { 4. } t2= { 1. 2. 3. } t3= { }
t1= { } t2= { 1. 2. 3. } t3= { 4. }

t1= { } t2= { 2. 3. } t3= { 1. 4. }
t1= { 2. } t2= { 3. } t3= { 1. 4. }

t1= { 1. 2. } t2= { 3. } t3= { 4. }
t1= { 1. 2. } t2= { } t3= { 3. 4. }

t1= { 2. } t2= { 1. } t3= { 3. 4. }
t1= { } t2= { 1. } t3= { 2. 3. 4. }

t1= { } t2= { } t3= { 1. 2. 3. 4. }

```

## TRAVAIL A FAIRE

- 1) Vérifier le déroulement de l'algorithme en parcourant les résultats d'exécution fournis.
  - 2) Lister les éléments prenant part à cet algorithme et écrire leur déclaration en C++. **Privilégier les collections d'éléments homogènes.**
  - 3) Corriger / compléter l'algorithme pour le rendre conforme aux réponses précédentes.
  - 4) Pour obtenir les affichages présentés dans le sujet, le programmeur ajouté à l'algorithme initial plusieurs appels d'une action « afficher les tours ».
    - a) Marquer d'une croix sur l'algorithme les endroits où cette action a été appelée.
    - b) Écrire la déclaration du sous-programme **afficherLesTours** correspondant
    - c) Écrire l'algorithme de **afficherLesTours**
  - 5) Écrire la déclaration C++ du sous-programme correspondant à l'algorithme « résoudre Hanoi »
  - 6) Nous aurons de déplacer des disques entre les tours. Pour ce faire :
    - a) Écrire la déclaration C++ du sous-programme **deplacerDisque**,
      - dont les paramètres sont :
        - **tourOrigine**, une tour
        - **tourDestination** une tour
      - et dont le but est de déplacer le sommet de **tourOrigine** vers la **tourDestination**. On supposera que le déplacement est possible
      - Écrire l'algorithme de ce sous-programme
  - 7) Action **preparerLesTours**.
    - b) En vous basant sur les résultats d'exécution fournis, indiquer le contenu de chaque tour après exécution de cette action.
    - c) Écrire la déclaration C++ d'un sous-programme **remplirTour**,
      - dont les paramètres sont :
        - **nbDisques**, le nombre de disques à empiler sur
        - **tourARemplir**, la tour recevant les disques
      - et dont le but est de remplir la tour avec le nombre de disques fournis, du plus grand (en bas) au plus petit (en haut) de la tour
    - d) Écrire l'algorithme de l'action **remplirTour**
    - e) Écrire l'algorithme de l'action **preparerLesTours**
  - 8) Action **deplacerPetit**.
    - a) Trouver une stratégie de résolution pour les actions **deplacerPetit**
    - b) Écrire en C++ la déclaration de ce sous-programme
    - c) Écrire l'algorithme de ce sous-programme
  - 9) Action **deplacerAutre** :
    - a) Trouver les déplacements qu'il est possible de faire dans cette action.
    - b) Trouver une stratégie de résolution pour les actions
    - c) Écrire en C++ la déclaration de ce sous-programme
    - d) Écrire l'algorithme de ce sous-programme
  - 10) Sous-programme **afficherTour**
    - a) Écrire la déclaration C++ d'un sous-programme **afficherTour**,
      - dont le paramètre est :
        - **tour** , la tour (pile) à afficher
      - et dont le but est d'afficher le contenu d'une tour selon le format suivant :

```
{ 2. 3. }
```

 pour une tour non vide  

```
{ }
```

 pour une tour vide
    - b) Écrire l'algorithme associé à la déclaration choisie.
- Inspirez-vous de l'exercice n°1 pour faire votre choix.**