R1.01 : Initiation au développement (partie 2)

Feuille TP n° 4

version 1

Utilisation de Piles

OBJECTIFS PEDAGOGIQUES :

3.- Apprendre les manipulations de base d'un Type Abstrait de Données Pile.
1.- Coder des algorithmes sous forme modulaire : création et utilisation de sous-programmes
2.- S'exercer à l'écriture progressive de programmes.

RESSOURCES A VOTRE DISPOSITION POUR REALISER CE TP :

- **`tp4.pdf`** : le présent sujet de TP
- **`ressourcesTP4.zip`** : une archive composée de
  - `pile.h, pile.cpp` : les fichiers implémentant le type `UnePile` ***à adapter à vos programmes***
  - `main.cpp` : le fichier dans lequel vous coderez l'exercice des tours de Hanoï

EXERCICES A CODER

Vous coderez les sous-programmes suivants :

- `initialiserTours()`
- `remplirTour()`
- `afficherTour()`
- `afficherTours()`
- `deplacerDisque()`

Une fois que vous les aurez TOUS codés, vous pourrez les **tester en décommentant** le corps du sous-programme **`resoudreToursHanoiManuel()`**.

Ce sous-programme vous permettra de jouer vous-mêmes à résoudre le problème des tours de Hanoï.

PREPARATION AU TRAVAIL

1. A l'aide de l'explorateur de fichiers, dans votre espace de travail de programmation vsCode (dossier **`r101_partie2`**), créer un dossier **`tp4`** pour accueillir les fichiers de cette feuille de TP.
2. Sur eLearn, télécharger l'archive **`ressourcesTP4.zip`**, et décompresser son contenu **dans** le dossier **`tp4`**.
3. Supprimer le fichier **`ressourcesTP4.zip`**

ADAPTATION DU MODULE PILE

Le module de gestion de piles doit gérer des éléments entiers, correspondant à la **taille** des disques placés dans chaque tour.

4. Vérifier / modifier le module Pile pour qu'il en soit ainsi.
5. Compiler.

ANALYSE DU FICHIER `main.cpp`

Analyse de la fonction `main()`

6. Après lecture du corps de la fonction `main()`, indiquer quel est son but.

Analyse du sous-programme `resoudreToursHanoiManuel()`

Ce sous-programme vous permettra de tester le bon fonctionnement des sous-programmes suivants :

- `initialiserTours()`
- `remplirTour()`
- `afficherTour()`
- `afficherTours()`
- `deplacerDisque()`

tout en jouant vous-mêmes aux tours de Hanoï.

Attention : pour jouer (et gagner), rappelez-vous que certains des sous-programmes développés ont des pré-conditions à respecter (par exemple, ne pas demander à déplacer un disque depuis une tour vide).

Analyse du fichier `main.cpp`

7. Trouver l'emplacement de la définition de la constante `NB_TOURS`. Commenter la notion de variable **globale**.

Des zones ont été identifiées pour y placer vos sous-programmes.

8. Trouver ces zones destinées à recevoir les sous-programmes :
   - Observateurs
   - Primitives modifiant 1 tour
   - Primitives modifiant plusieurs tours
   - Affichages
   - Algorithmes de résolution du problème des tours de Hanoï
   - Boîte à outils pour resolutionToursHanoiManuel()
   - Boîte à outils pour resolutionToursHanoiAutomatique() (celui de la feuille de TD n°5)

Les emplacements des sous-programmes que vous avez à coder aujourd'hui sont suggérés dans le fichier.

9. Trouvez-les.

CODAGE DES SOUS-PROGRAMMES

10. Coder le corps de chacun des sous-programmes :

- `initialiserTours()`
- `remplirTour()`
- `afficherTour()`
- `afficherTours()`
- `deplacerDisque()`

de manière incrémentale, en compilant régulièrement.

TESTER LES LES SOUS-PROGRAMMES

11. Décommenter le **corps** du sous-programme `resoudreToursHanoiManuel()`
12. Compiler
13. Jouer pour tester vos sous-programmes.
    Si vous respectez l'algorithme proposé sur la feuille de TD, à savoir :
    Répéter
        déplacerPetit (de t1→ t2 → t3 → t1…),
        puis déplacerAutre disque que le petit quand cela est possible
    finRépéter

    Vous devriez obtenir le même résultat d'exécution que celui montré sur la feuille de TD n°5.

RAPPELS

Avant de coder :

- Faire un algorithme

Lors du codage : appliquer toutes les recommandations vues dans la première partie de R1.01-Initiation au développement (partie 1).