TP7 : Pile et file

Les objectifs de ce TP sont les suivants :

- Faire fonctionner ensemble différentes structures de données dynamiques dans un même programme,
- Savoir écrire un Makefile avec plus d'unités de compilation,
- Mettre en œuvre la notion d'abstraction à travers l'implémentation des services fondamentaux offerts par des modules Pile et File.

Dans ce TP, vous allez écrire un module Pile (basé sur le module TableauDynamique du TP5), et un module File (basé sur le module Liste du TP6). Vous devez avoir fini l'exercice 3 du TP5 et l'exercice 2 du TP6 de sorte à obtenir des modules TableauDynamique et Liste avec toutes les fonctionnalités de base, et correctement testés.

Exercice 1 : Classe File

a. Dans votre répertoire LIFAPSD, créez un répertoire TP7. Placez-y les fichiers ElementL.h et ElementL.cpp que vous trouverez sur le site de l'UE. Observez le contenu de ces fichiers : au lieu de stocker des entiers dans les listes, nous allons stocker des adresses (pointeur générique : `void *`).

Cela veut donc dire que l'on insèrera un élément (par exemple un entier) en passant en paramètre de `enfiler` un pointeur sur cet élément (pouvant être soit sur la pile soit sur le tas). Ex. : `mafile.enfiler(new int(1));`

De même lorsque l'on souhaite récupérer un élément depuis la file, nous allons récupérer par défaut un `void*` que l'on va ensuite convertir dans le type souhaité. Ex. : `int * elem = (int*) mafile.premierDeLaFile();`

b. Copiez dans votre répertoire TP7 vos fichiers Liste.h et Liste.cpp du TP6.

c. Toujours dans votre répertoire TP7, ajoutez le fichier File.h qui se trouve sur le site de l'UE. Ouvrez-le : vous verrez que la classe File contient simplement une donnée membre de type Liste et les opérations spécifiques à File : enfiler, défiler, etc. Créez un nouveau fichier File.cpp et écrivez le code des fonctions membres annoncées dans File.h. **Chacune de ces fonctions s'écrit en une ligne uniquement**.

C'est un exemple d'abstraction et d'encapsulation : la « sur-couche » File, par dessus le module Liste, permet à l'utilisateur du module File de la voir comme une boîte noire, simple à utiliser, qui ne propose que les services permis sur une File. L'utilisateur du module File n'a pas à savoir si elle est implémentée sous forme d'une liste chaînée ou d'un tableau dynamique ou autre. Idéalement, il ne voit que les services proposés dans le fichier File.h et ne risque pas d'effectuer des opérations illégales sur une File, comme une insertion en plein milieu par exemple.

d. Placez dans votre répertoire TP7 une copie du Makefile de votre TP6 et complétez-le pour prendre en compte les deux unités de compilation (Liste et File). Attention à bien mettre à jour la liste des .h dans les listes de dépendances : on rappelle que pour une ligne commençant par 'fichier.o:', il faut indiquer le fichier .cpp correspondant et tous les .h inclus dans ce .cpp (et seulement ceux-là). Ecrire un programme principal qui teste toutes les fonctions de la classe File, compilez et corrigez votre code jusqu'à ce qu'il fonctionne.

Exercice 2 : Classe Pile

a. Copiez dans votre répertoire TP7 vos fichiers TableauDynamique.h et TableauDynamique.cpp du TP5.

b. Placez dans votre répertoire TP7 les fichiers ElementTD.h, ElementTD.cpp et Pile.h que vous trouverez sur le site de l'UE. Vous observerez que le type Pile contient simplement une donnée membre de type TableauDynamique et les opérations spécifiques à Pile : empiler, dépiler, etc. Créez un nouveau fichier

Pile.cpp et écrivez le code des fonctions membres annoncées dans Pile.h. **Chacune de ces fonctions s'écrit en une ligne uniquement**.

c. Complétez le Makefile du TP5 pour prendre en compte les deux unités de compilation (TableauDynamique et Pile). Ecrire un programme principal qui teste toutes les fonctions de la classe Pile, compilez et corrigez votre code jusqu'à ce qu'il fonctionne.