TP no4 : Structure Abstraite /Concrète

1 La Structure de Données Abstraite (SDA)

On définit ici une cartouchiere comme une structure de données indicée sur laquelle agissent les fonctions suivantes (les paramètres soulignés sont en fait des données modifiées—passées par référence) :

```
init : int → cartouchiere
// PRE la cartouchiere n'est pas encore initialisée
// → initialise la cartouchiere avec ce nombre d'emplacements

estLibre : cartouchiere × int → booléen
// PRE la cartouchiere a été initialisée
// → FAUX si et seulement si l'emplacement à ce numéro est déjà/encore occupé

ajout : cartouchiere  × T → cartouchiere × int
// PRE la cartouchiere n'est pas pleine
// → ajoute la valeur dans une emplacement libre de la cartouchiere (renvoie son numéro)

vider : cartouchiere × int → cartouchiere
// PRE le numéro correspond à un emplacement non vide
// → vide l'emplacement dont le numéro est donné

valeurEn: cartouchiere × int → T
// PRE le numéro correspond à un emplacement non vide
// → donne la valeur stockée à l'emplacement dont le numéro est donné
```

L'implémentation de cette SDA va donc comporter un fichier de header avec le profil de ces fonctions (fourni dans les fichiers disponibles sur Madoc).

2 Contexte de programmation

Pour ce projet, vous allez implémenter au moins deux Structures de Données Concrètes (SDC) correspondant à cette SDA. Toutes ces SDC seront contenues dans des dossiers différents, mais avec les mêmes noms de fichier, et contenant les mêmes noms de fonctions. Le programme principal devra pouvoir être compilé indifféremment avec l'une ou l'autre. Les résultats à l'exécution devront dans chaque cas être cohérents avec la spécification (mais pas nécessairement exactement identiques).

Chaque SDC fait référence au type générique des éléments et à la SDA et devra être compilable séparément en se plaçant dans son dossier et en tapant

```
gpp cartouchiere.cpp -c -I ../
```

Pour compiler le programme principal, un petit fichier batch est fourni ; le lancer avec le nom du sous-dossier contenant l'implémentation choisie, par exemple

```
./construire C1
```

Remarque : il existe de meilleures méthodes pour automatiser le processus de compilation (makefile e.g.), il existe aussi dans le langage des façons plus propres de gérer la généricité. On se contentera dans ce projet de ce qui est proposé.

3 Les Structures de Données Concrètes

Vous devez développer au minimum deux SDC. La première utilisera un (ou plusieurs) tableau statique dimensionné par la procédure d'initialisation, la seconde sera un chaînage de couples (valeur, numéro).

Ces SDC doivent fonctionner au moins pour les types habituels de T_ELT (float, int, string) mais vous pouvez étudier (et discuter) le cas de structures ou de pointeurs vers des structures.

4 Le programme principal

Une telle structure de données est utile par exemple pour affecter des ressources informatiques : gestion de la mémoire, ports de communication. Le programme que vous devez réaliser doit simuler un hôtel : lorsqu'un client arrive, il est affecté à une chambres libre et le numéro de chambre lui est fourni ; lorsqu'il quitte l'hôtel, sa chambre est libérée, et devient disponible pour un autre client. Votre programme doit proposer un menu permettant d'accueillir un nouveau client (défini par son nom, une chaîne de caractères) , de libérer une chambre, d'afficher la liste des clients de l'hôtel, d'afficher le taux d'occupation,…

5 Travail demandé

Il est conseillé de travailler progressivement :

a) Développez une première SDC et complétez au fur et à mesure un petit programme principal pour tester ses fonctionnalités ; rédigez en parallèle un document d'accompagnement précisant les décisions prises et les difficultés rencontrées. Le code doit comporter les PRE/POST conditions importantes ainsi que les ordres de grandeur des coûts des fonctions. Le fichier SDAcartouchiere.h ne doit pas être modifié.

b) Développez ensuite de même une seconde SDC ; rédigez en parallèle un document d'accompagnement similaire. Vérifiez régulièrement que vous pouvez compiler et exécuter le programme de test indifféremment avec l'une ou l'autre des SDC.

c) Développez enfin le programme de gestion de chambres ; il doit pouvoir compiler avec l'une ou l'autre des SDC. Dans cette étape, vous pouvez proposer des ajouts raisonnables (et raisonnés) à la SDA et les implémenter dans les SDC, mais le rapport devra pour chaque fonction ajoutée présenter un algorithme équivalent utilisant uniquement la SDA donnée et justifier l'intérêt de cet ajout.

d) Le rapport doit en particulier comporter une discussion des choix effectués pour chaque SDC (structure triée ou non, informations mémorisées ou pas—par exemple le nombre d'élément déjà entrés, un pointeur vers la queue —, chaînage simple ou double, etc.) mais aussi un comparatif des deux SDC implémentées en termes de place mémoire et de coût d'exécution.

6 Annexe

Extrait du fichier genericite.h définissant le type des éléments (à modifier pour tester sur d'autres types) :

```
typedef float T_ELT;
```

Extrait du fichier SDAcartouchiere.h donnant la traduction C++ des signatures des fonctions :

```
void cart_init(cartouchiere & c, int n );
bool estLibre(const cartouchiere & c, int p);
int ajout(cartouchiere & c,T_ELT t);
void vider(cartouchiere & c,int p);
T_ELT valeurEn(const cartouchiere & c,int p);
```

Remarques :

- d'autres signatures auraient pu être choisies, vous pouvez en parler dans le rapport,
- les fonctions agissent par modification sur la structure, elle ne créent pas de copie.