Compte-Rendu TP C++ #1

Binôme B3408
Renault Benoit, Espeute Clément

1 Spécifications de la classe <Collection>

1.1 Présentation générale

<Collection> a pour rôle de gérer le stockage, la modification et l'affichage d'un ensemble de *nombres entiers*. Le stockage est réalisé dans un tableau d'entiers (int) géré de façon dynamique simple, c'est à dire que l'on a également deux attributs de type entier non-signé, l'un servant à connaître le nombre d'éléments actuellement présents dans le tableau, l'autre servant à connaître la taille allouée au tableau. Le cahier des charges ne l'exigeant pas, les redondances sont acceptées et les valeurs ne sont pas triées.

1.2 Présentation des méthodes

Constructeur n°1

Créé une collection pouvant accueillir un nombre entier non-signé donné d'éléments sans avoir besoin d'être redimensionnée, et alloue la mémoire nécessaire au tableau.

Constructeur n°2

Créé une collection pouvant accueillir un nombre entier non-signé donné d'éléments sans avoir besoin d'être redimensionnée, et alloue la mémoire nécessaire au tableau. Les valeurs du tableau d'entiers quelconque dont le pointeur est passé en paramètre sont ensuite ajoutées à la collection (part du contrat : il est à la charge de l'utilisateur de donner une taille égale à celle du tableau passé).

Méthode Afficher

Imprime sur la sortie standard les éléments de la collection dans l'ordre d'apparition du tableau, avec une virgule entre chaque élément sur la sortie standard et deux retours à la ligne à la fin de l'affichage. Si en mode de déboguage (`#define DEBUG`), affiche en plus le nombre d'éléments et la taille allouée avent l'affichage des éléments.

Méthode Ajouter

Ajoute la valeur entière donnée dans un élément inséré à l'extrêmité du tableau et fait appel à la fonction d'ajustement si l'insertion implique un dépassement de la taille actuellement allouée au tableau. [1]

[1] Tout ajout ou suppression du nombre d'éléments contenus dans le tableau implique bien sûr une mise à jour de l'attribut associé.

Méthode Retirer

Supprime un nombre entier d'occurences donné d'une valeur entière donnée. Si le nombre d'occurences à retirer donné est négatif, on supprime toutes les occurrences de la valeur entière donnée. La taille du tableau est ajustée après l'opération.

Méthode Ajuster

Modifie la taille allouee au tableau et le réalloue dynamiquement à une valeur entière non-signée donnée en paramètre. Le redimensionnement est refusé si la taille demandée est inférieur au nombre actuel d'éléments et renvoie un code de retour `ERR_TAILLE`.

Méthode Reunir

Ajoute les éléments de la collection donnée après ceux de la courante en réajustant la taille allouee du tableau de la courante si celle-ci n'est actuellement pas assez grande pour les accueillir.

Méthode Destructeur

Effets : Désallocation du tableau. Détruit la collection courante et libère la mémoire allouée au tableau associé.

2 Les tests

Afin de vérifier que chaque méthode notre classe fonctionne, on réalise des tests unitaires sur chacune d'entre elles. La méthode Afficher à été modifiée pour pouvoir donner la taille allouée du tableau et le nombre d'éléments a l'intérieur de celui-ci. Cet affichage peut être désactivé en enlevant le mot clef `#define DEBUG` à l'intérieur de `Collection.h`.

Par ailleurs, afin de rendre les tests plus lisibles dans la console, les textes indiquant les différentes étapes des tests sont imprimés en gras. Si la console ne supporte pas les caractères d'échappement permettant la mise en forme des textes affichés, ils peuvent être désactivé en retirant le mot clef `#define BOLD_USAGE` à l'interieur de `main.h`.

2.1 Tests pour le deuxième constructeur

`testConstructeur2_X()` sont les méthodes qui effectuent le test pour le second constructeur.

`testConstructeur2_1()` Teste le cas de base où l'on passe un tableau rempli arbitrairement en paramètre du constructeur de la Collection (ainsi que sa taille). Le tableau initial est :

```
int leTableau[] = {100,200,300,400,500,600,700,800,900,999};
```

Résultat attendu : La Collection est bien initialisée et elle contient tous les éléments du tableau, et à la même taille. (`allouée = nbElements`)

Résultat obtenu :

```
NbElements : 10
Alloue : 10
100,200,300,400,500,600,700,800,900,999
```

`testConstructeur2_2()` Teste le cas où l'on donne en paramètre un tableau vide.

Résultat attendu : La Collection est bien initialisée et elle est vide. `allouee` et `nbElements` est de 0.

Résultat obtenu :

```
NbElements : 0
Alloue : 0
```

`testConstructeur2_3()` Teste le cas où l'on appelle le constructeur avec en paramètre un tableau et une taille n plus petite que celle du tableau. Le tableau initial est le même que pour le premier test, et on donne en nombre d'élément 5.

Résultat attendu : La Collection est bien initialisée, et elle est remplie avec les n premiers éléments du tableau passé en paramètre.

Résultat obtenu :

```
NbElements : 5
Alloue : 5
100,200,300,400,500
```

Note : On ne teste pas le cas où l'on appelle le constructeur avec un tableau et un nombre d'éléments supérieur au vrai nombre d'éléments contenu dans le tableau, car le contrat dans le manuel du constructeur interdit ce genre d'appel. L'erreur ne viendra pas de la programmation du constructeur, mais de l'utilisateur.

2.2 Tests pour `Ajouter`

Le test pour la méthode ajouter comporte 2 tests :

`testAjouter1()` Ajoute simplement un élément dans la Collection vide avec une taille initiale de 5 (donc pas de redimensionnement nécessaire). On affiche l'état du tableau avant et après l'ajout.

Résultat attendu : Le tableau ne comporte que l'élément ajouté. `nbElement` vaut 1 et `alouee` vaut toujours 5. Résultat obtenu :

```
===== Collection initiale =====
NbElements : 0
Alloue : 5
===== Collection après ajout =====
NbElements : 1
Alloue : 5
5
```

`testAjouter2()` Ajoute 20 éléments dans une Collection initialement vide. (Donc redimensionnement nécessaire).

Résultat attendu : Le tableau est rempli avec les 20 éléments dans l'ordre dans lequel ils ont été insérés. `alouee = nbElement = 20`.

Résultat obtenu :

```
===== Collection de base : =====
NbElements : 0
Alloue : 0
===== Collection après l'ajout de 20 éléments: =====
NbElements : 20
Alloue : 20
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19
```

2.3 Tests pour Retirer

Pour tester la méthode retirer, on remplit au préalable la Collection d'un tableau arbitraire, puis on tente de retirer des éléments. On affiche à chaque fois la Collection avant et après l'opération.

Pour les 4 premiers tests, la collection est initialisé avec le tableau suivant :

```
int tabBase[] = {1,42,1,42,8,4,42,42,42,1,8,9,42,42,8,7,3,42,8};
```

On à alors :

testRetirer1() Retire un élément (ici 42) d'une valeur donnée de la collection.

Résultat attendu : L'élément est supprimé et la collections à bien été redimensionnée à la taille initiale - 1.

Résultat obtenu :

```
========== Collection initiale : =====
NbElements : 19
Alloue : 19
1,42,1,42,8,4,42,42,42,1,8,9,42,42,8,7,3,42,8
===== On retire 1 élément 42 =====
NbElements : 18
Alloue : 18
1,8,1,42,8,4,42,42,42,1,8,9,42,42,8,7,3,42
```

Note : le premier élément 42 de la liste à été supprimé, et le dernier élément de la liste (le 8) à pris ça place comme spécifié dans la spécification de la méthode Retirer().

testRetirer2() Retire n éléments d'une valeur donnée de la collection.

Résultat attendu : L'élément est supprimé et la collections à bien été redimensionnée à la taille initiale - n.

Résultat obtenu :

```
===== Collection initiale : =====
NbElements : 19
Alloue : 19
1,42,1,42,8,4,42,42,42,1,8,9,42,42,8,7,3,42,8
===== On retire 4 éléments 42 =====
NbElements : 15
Alloue : 15
1,8,1,3,8,4,7,42,42,1,8,9,42,42,8
```

testRetirer3() Retire tous les éléments d'une valeur donnée du tableau.

Résultat attendu : Tout les éléments de la valeur donnée sont supprimés et la collections à bien été redimensionnée correctement.

Résultat obtenu :

```
===== Collection initiale : =====
NbElements : 19
Alloue : 19
1,42,1,42,8,4,42,42,42,1,8,9,42,42,8,7,3,42,8
===== Collection après Retirer(42,-1) =====
NbElements : 11
Alloue : 11
1,8,1,3,8,4,7,8,9,1,8
```

testRetirer4() Tente de retirer une valeur non présente dans le tableau.

Résultat attendu : Le tableau ne change pas, et il n'y pas d'erreur produite.

Résultat obtenu :

```
===== Collection initiale : =====
NbElements : 19
Alloue : 19
1,42,1,42,8,4,42,42,42,1,8,9,42,42,8,7,3,42,8
===== Collection après Retirer(100,-1) =====
NbElements : 19
Alloue : 19
1,42,1,42,8,4,42,42,42,1,8,9,42,42,8,7,3,42,8
```

testRetirer5() Retire procéduralement touts les éléments du tableau.

Résultat attendu : Le tableau est vide, `nbElement = alouee = 0`

Résultat obtenu :

```
===== Collection initiale : =====
NbElements : 19
Alloue : 19
1,9,1,9,8,4,9,9,9,1,8,9,9,9,8,7,3,9,8
===== On retire tous les éléments de la Collection =====
NbElements : 0
Alloue : 0
```

2.4 Tests pour Réunir

Pour vérifier le bon fonctionnement de la méthode `Reunir`, on procède à 3 test différents. On affiche à chaque fois les Collections initiales et la Collection correspondant à la réunion des deux.

testReunir1() Crée 2 Collections vides et les réunit.

Résultat Attendu : Aucune différence entre la première collection et la collection réunie.

Résultat obtenu :

```
===== Collection 1 =====
NbElements : 0
Alloue : 10
===== Collection 2 =====
NbElements : 0
Alloue : 20
===== Réunion des 2 =====
NbElements : 0
Alloue : 10
```

testReunir2() Crée 2 Collections et les réunit. Les Collections sont telles que la réunion n'implique pas de redimensionement de la première collection.

Résultat Attendu : La collection réunie contient les éléments de la première suivie des éléments de la seconde. Sa taille `alouee` est égale à celle de la première collection car il n'y a pas eu de redimensionement.

Résultat obtenu :

```
===== Collection 1 =====
NbElements : 15
Alloue : 30
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14
===== Collection 2 =====
NbElements : 5
Alloue : 10
10,20,30,40,50
===== Réunion des 2 =====
NbElements : 20
Alloue : 30
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,10,20,30,40,50
```

`testReunir3()` Crée 2 Collections et les réunit. La première collection est remplie ce qui implique qu'un redimensionement sera nécessaire lors de la réunion des Collections.

Résultat Attendu : La collection réunie contient les éléments de la première suivie des éléments de la seconde. Sa taille `alouee` est égale à `nbElement`

Résultat Obtenu :

```
===== Collection 1 =====
NbElements : 15
Alloue : 15
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14
===== Collection 2 =====
NbElements : 10
Alloue : 15
10,20,30,40,50,60,70,80,90,100
===== Réunion des 2 =====
NbElements : 25
Alloue : 25
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,10,20,30,40,50,60,70,80,90,100
```