

TP1: Une première classe

B3111: Edern HAUMONT et Nicolas SIX

Mercredi 21 octobre 2015

1 Introduction

La classe BoolContainer gère une collection dynamique de booléens non triée. Cette collection a une taille spécifiée à sa création, cependant elle peut être modifiée à la demande ou optimisée automatiquement. La classe est dotée de méthodes d’affichage, d’ajout et de suppression d’items, de réunion d’une autre collection de booléens et de modification de capacité. En interne, les booléens sont stockés dans un tableau statique de booléens. Celui-ci présente l’avantage de prendre peu de place en mémoire et de permettre un accès rapide aux éléments. Cependant, cela impose de recréer un tableau et de copier recopier tous les éléments du tableau un à un à chaque fois que sa taille est modifiée. Les attributs et méthodes de la classe sont spécifiés ci-après.

2 Atributs

La classe BoolContainer utilise trois atribus privé pour gérer ses données :

```
unsigned int  tailleUtilisee;  
unsigned int  tailleDispo;  
bool *tab;
```

Cela lui permet de connaitre la quantiter de donnée actuellement stocker mais aussi d’avoir une zone de stockage par l’intermédiaire du pointeur.

3 Méthodes publiques

3.1 Constructeur par défaut

```
BoolContainer(unsigned int nouvelleTaille = DEFAULT_CONTAINER_SIZE);
```

Prends en parametre optionnel la taille de la nouvelle collection. La taille prise par défaut est une constante définie dans le .h de la classe.

3.2 Constructeur avec tableau d’initialisation

```
BoolContainer(bool nouveauTab[], unsigned int nouvelleTaille);
```

Prends en parametre un tableau statique de booleans qui sera copie et sa taille.

Contrat : nouveauTab est de taille supérieur ou égale à nouvelleTaille (si sa taille est supérieure à nouvelle taille seuls les éléments des cases entre 0 et nouvelleTaille exclue seront copié dans le tableau).

3.3 Destructeur

```
virtual ~BoolContainer ( );
```

Détruit le tableau.

3.4 Afficher

```
int Afficher(bool afficherCarateristique=true, bool afficherContenue=true) const;
```

Peut afficher la capacité utilisée et disponible suivie du contenu du BoolContainer. Si le premier argument est vrai alors il y aura affichage d'une ligne avec les caractéristiques du tableau. Si le second est vrai alors il y aura affichage du contenu du tableau. Par défaut : affiche les deux blocs.

3.5 Ajouter

```
int Ajouter(bool b);
```

Prends en paramètre le booleen qui sera ajoute a la fin de la liste.

3.6 Retirer

```
int Retirer(unsigned int debut, unsigned int longueur = 1);
```

Prends en paramètre l'index e partir auquel la suppression aura lieu et en parametre optionnel, la longueur de l'intervalle d'index sur lequel les suppressions auront lieu.

3.7 Ajuster

```
int Ajuster(unsigned int nouvelletaille);
```

Prends en paramètre la taille de collection désirée (supérieure à la taille utilisée actuellement).

3.8 Réunir

```
int Reunir(const BoolContainer & boolContainerBis);
```

Prends en paramètre une autre collection de booléens qui sera ajoutée à la fin de la collection courante.

4 Codes d'erreur

Nous avons choisi de faire renvoyer un code entier par nos fonctions de façon à détecter plus facilement certaines erreurs d'exécution via des codes :

— **const int** ERROR_NON_INTUITIVE_ADJUSTMENT = 100;

Ce code d'erreur est renvoyé par la méthode Ajuster si la nouvelle taille demandée est inférieure à la taille actuellement utilisée par la collection.

— **const int** ERROR_RESIZE_FAILED = 101;

Échec de redimensionnement du tableau.

— **const int** ERROR_OUT_OF_BOUNDARY = 102;

Ce code d'erreur est renvoyé par la méthode Retirer si un ou des éléments dont on demande le retrait ne se trouvent pas dans la collection, c'est-à-dire au-delà de tailleUtilisee.

5 Tests unitaires

Pour des raisons d’encapsulation le programme ne peut pas effectuer la plupart des tests par lui-même. De fait la plupart des tests se font par contrôle visuel : l’utilisateur ayant lancé le programme de test doit indiquer lui-même si les tests ont réussie ou échouer.

5.1 Test de la méthode Ajouter

test visuel :

La fonction crée un BoolContainer vide, puis utilise la méthode Ajouter pour y insérer des éléments. La méthode Afficher est appelée. L’utilisateur n’a qu’à comparer le résultat de Afficher à ce qui était attendu.

5.2 Test de la méthode Retirer

test visuel :

La fonction crée un BoolContainer et le remplit. Elle supprime ensuite un élément puis un groupe d’éléments et permet à chaque fois à l’utilisateur de vérifier que les bons éléments ont été retirés.

5.3 Test de la méthode Réunir

test visuel :

La fonction crée deux BoolContainer. Elle les affiche. Puis elle appelle la méthode réunir sur le premier tableau avec le second comme argument. Enfin elle affiche le résultat et l’utilisateur vérifie la correspondance.

5.4 Test d’overflow

test automatisé :

Ce test vérifie qu’il n’y a pas de problème lors de l’insertion dans un tableau plein. Pour cela il crée un tableau de taille un. Puis ajoute des éléments un à un pour essayer de supprimer le même nombre d’élément que celui qui a été ajouté. Le test ne vérifie pas l’intégrité des données mais permet de s’assurer que les tableaux se remplissent dans ce cas sans perte d’éléments.

Le test étant automatisé il effectue ses tests sur un grand nombre de tableaux. Même si cela ne semble pas nécessaire, nous nous assurons ainsi d’une certaine stabilité du processus sans que l’utilisateur ne puisse voire la différence.

5.5 Test de la méthode Ajuster

test automatisé :

Ce test génère un tableau et applique la méthode ajuster dessus. En fonction des paramètres employés le test vérifie qu’il n’y a pas de comportement imprévue : acceptation d’un redimensionnement à une taille inférieure à la taille occupé par les données ou refus d’un redimensionnement alors que le nombre de donnée le permettait.