420-JWA-BT Algorithmique et programmation
Hiver 2018

Laboratoire 6 (2,5%)

Votre dernier laboratoire vous servira pour pratiquer vos notions de pointeurs et d'allocations sur le tas. Votre objectif est d'implémenter votre propre version de la classe **`std::vector`**!

Votre classe doit implémenter toutes les méthodes spécifiées dans le fichier **`Vector.h`**. Il est permis de modifier ce fichier mais vous ne pouvez pas modifier les méthodes déjà présentes. Votre vecteur ne doit pas effectuer de copies d'éléments, les éléments doivent être stockés sous forme de pointeurs. Votre vecteur est donc uniquement responsable de libérer la mémoire qu'il a alloué lui-même.

Votre vecteur devrait avoir initialement une taille de 0 et une capacité de 1. Lorsque la capacité est insuffisante, vous devrez créer un nouveau tableau avec un capacité doublée.

Vous devez aussi vous assurer de ne jamais avoir de fuite de mémoire. Vous devez libérer tous vos tableaux une fois remplacés et le tableau actuel dans votre destructeur.

Voici une description de chaque méthode à implémenter:

- **`int Vector::size()`**
  Cette méthode retourne le nombre d'éléments actuellement dans le vecteur.

- **`int Vector::capacity()`**
  Cette méthode retourne la capacité réelle du tableau actuel du vecteur.

- **`void Vector::add(const std::string& element)`**
  Cette méthode ajoute un nouvel élément à la fin du tableau. Si la capacité est insuffisante, vous devez doubler la taille du tableau et copier les anciens éléments dans le nouveau tableau, puis détruire l'ancien tableau.

- **`void Vector::insert(const int position, const std::string& element)`**
  Cette méthode insère un élément à la position donnée. Tous les éléments suivants doivent être déplacés pour lui donner une place. Si la capacité est insuffisante, vous

devez doubler la taille du tableau et copier les anciens éléments vers le nouveau tableau, puis détruire l'ancien tableau.

- **`const std::string& Vector::get(const int position)`**
  Cette méthode retourne une référence à l'élément présentement à la position donnée.

- **`void Vector::remove(const int position)`**
  Cette méthode efface un élément à la position donnée. Tous les éléments suivants doivent être déplacés d'un espace pour combler le trou. La capacité ne devrait pas être affectée.

- **`void Vector::clear()`**
  Cette méthode efface tous les éléments du vecteur. La capacité ne devrait pas être affectée.

Le fichier **`Laboratoire6.cpp`** fourni effectue une batterie de tests pour vérifier votre implémentation. Ceux-ci devraient passer sans erreurs.

Évaluation

Vous serez évalués sur les points suivants:

- **Initialisation (5 points):**
  - Votre vecteur doit débuter avec une taille de 0 et une capacité de 1.
- **Destruction (5 points):**
  - La mémoire du tableau interne doit se libérer à la destruction.
- **Aucune copie (5 points):**
  - Les **`std::string`** doivent être gardées sous forme de pointeurs.
- **Méthode `size` (5 points)**
- **Méthode `capacity` (5 points)**
- **Méthode `add` (10 points)**
- **Méthode `insert` (15 points)**
- **Méthode `get` (5 points)**
- **Méthode `remove` (10 points)**
- **Méthode `clear` (5 points)**

Échéancier

Vous devrez livrer vos fichiers CPP et H, dans un ZIP, par Léa mercredi le 18 avril avant minuit.