Travaux dirigés C++ n°7

Informatique

—IMAC 2e année—

La STL

Au cours de ce TP vous apprendrez à utiliser la Standard Template Library.
Il est conseillé de se référer à la documentation en ligne :

- http://cplusplus.com/reference
- http://cplusplus.com/reference/stl

▸ Exercice 1. *Les vectors*

1. *Quelle est la complexité d'accès à un élément d'un* `std::vector` *?*
2. *Déclarez un* `std::vector<int>` *vide et faites une boucle dans laquelle à chaque nouvelle itération, vous lui ajoutez un élément à l'aide de la méthode* `push_back()`*. À chaque itération, affichez la taille du vecteur (le nombre d'éléments qu'il exprime) avec la méthode* `size()` *ainsi que la taille mémoire réellement allouée à l'aide de la méthode* `capacity()`*. Qu'observe-t-on?*
3. *Quelle est la complexité d'ajout d'un élément à un élément d'un* `std::vector` *?*
4. *Vérifiez que la méthode* `shrink_to_fit()` *vous permet d'ajuster la mémoire allouée à la taille des données exprimées.*
5. *Faites un test avec la méthode* `reserve()`*. Est-ce que vous devez quand même faire des* `push_back()` *pour insérer des éléments? En est-il de même si vous utilisez le constructeur spécifiant la taille du vecteur?*

▸ Exercice 2. *Les itérateurs*

1. *Utilisez un* `std::vector<int>::iterator` *pour parcourir et afficher le contenu du* `std::vector` *précédemment créé.*
2. *Faites une fonction d'affichage de votre vecteur avec le propotype suivant :*
   `void afficheVector(const std::vector<int> &vec)`
   Quel changement devez vous opérer par rapport à l'affichage de la question précédente?
3. *Faites mes mêmes boucles sans les itérateurs, en utilisant la forme compacte du genre* `for(auto e : vec)` {...}
   mais en utilisant une référence constante sur chaque élément du vecteur.

▸ **Exercice 3.** ***Les algorithmes***

1. *Triez votre* `std::vector<int>` *à l'aide de la fonction* `std::sort`*.*
2. *Triez votre* `std::vector<int>` *à l'aide de la fonction* `std::sort` *selon les critères de tri suivants (les 3 en même temps) :*
   - *si a est pair et b impaire, alors a est classé avant b.*
   - *si a et b sont pairs, alors le plus petit des deux est classé en premier.*
   - *si a et b sont impaires, alors le plus petit des deux est classé en premier.*

   Soit en gros d'abord les nombres pairs triés, puis les nombres impaires triés.
3. *À l'aide de la fonction* `count`*, comptez le nombre d'occurrences de la valeur 7 dans ce vecteur*
4. *Créez un second vecteur* `vec2` *de la même taille que votre vecteur initial et calculez leur produit scalaire à l'aide de la fonction* `std::inner_product`*.*

▸ **Exercice 4.** ***Les listes***

Les listes sont implémentées comme des listes chaînées, cela permet des insertions rapides au début et à la fin de la liste. Grâce aux itérateurs, des éléments peuvent être insérés au milieu de listes.
Une liste prend en charge un certain nombre d'opérations :

- `merge()` *: Fusionner les listes*
- `reverse()` *: Inverser l'ordre des éléments*
- `unique()` *: supprimer les doublons d'une liste triée*

1. *Définissez une liste philo de philosophes : Platon, Aristote, Descartes et Kant.*
2. *Définissez une deuxième liste math de mathématiciens : Gauss, Laplace, Poincaré et Descartes.*
3. *Affichez les deux listes triées (le* `sort()` *de* `list`*).*
4. *Fusionnez les deux listes et stocker le résultat dans une nouvelle liste all (vous pouvez utiliser* `merge`*).*
5. *Supprimez les répétitions dans la liste all avec* `unique()`*.*
6. *Inversez l'ordre de la liste all puis l'afficher.*

Exécution :

```
philo : Platon Aristote Descarte Kant
math : Gauss Laplace Poincaré Descartes
philo triée : Aristote Descartes Kant Platon
math triée : Descartes Gauss Laplace Poincaré
all : Aristote Descartes Descartes Gauss Kant Laplace Platon Poincaré
all sans répétitions : Aristote Descartes Gauss Kant Laplace Platon Poincaré
all inversée : Poincaré Platon Laplace Kant Gauss Descartes Aristote
```

▸ **Exercice 5.** ***Les maps***

1. *Créez une* `std::map<std::string,int>` *associant une clé correspondant à un nom de département (de type* `std::string`*) avec un entier correspondant à son numéro de département. Ajoutez à vote map les départements suivants :*

   - *Aveyron : 12*
   - *Côtes d'Armor : 22*
   - *Seine-et-Marne : 77*
   - *Haute-Savoie : 74*

   Pour cela, vous utiliserez la méthode `insert` *associé à une* `std::pair<std::string, int>` *représentant le département à ajouter à la map.*

2. *Parcourez et affichez les éléments de votre map.*