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/ et notamment http://cplusplus.com/reference/stl/.

▸ **Exercice 1.** *Le conteneur **vector***

Objectifs : La construction et l'affichage des propriétés élémentaires d'un vecteur.

Outils : empty(), size(), max_size(), push_back().

1. *déclarer d'un vecteur encore vide contenant des entiers*
2. *afficher si le vecteur est vide en testant avec empty()*
3. *afficher la taille du vecteur*
4. *afficher la taille maximale d'un vecteur d'entiers*
5. *introduire quelques éléments dans le vecteur*
6. *afficher la nouvelle taille*
7. *afficher le contenu du vecteur*

▸ **Exercice 2.** ***vector et swap()***

La fonction swap() permet d'échanger les contenus de deux vecteurs. Utiliser cette fonction pour échanger deux vecteurs v1 et v2 de type std :: vector < std :: string > contenant "Test" et "Swap" respectivement.

▸ **Exercice 3.** *Algorithmes: sort, count*

La bibliothèque standard ne se réduit pas à la définition de conteneurs. Elle définit également un jeu de fonctions appelées algorithmes. Pour ces algorithmes, l'accès aux données contenues dans un conteneur s'effectue exclusivement par le biais d'itérateurs.

Ecrire un code qui permet de :

1. *définir un vecteur v de 20 entiers*
2. *remplir le vecteur v de manière aléatoire au moyen de la fonction rand() avec des valeurs entre 0 et 20*
3. *afficher ce vecteur*
4. *le trier*
5. *l'afficher à nouveau*
6. *compter le nombre de fois la valeur 7 existe dans ce vecteur*

▸ **Exercice 4.** *Le conteneur **deque***
Objectifs : comprendre le fonctionnement du conteneur deque.
Outils : utiliser push_front(), pop_back()

Le conteneur deque (doubly ended queue) est assez similaire au conteneur vector, à ceci près que sa finalité de donner un rôle équivalent au début et à la fin de la suite d'éléments qu'il contient.

1. *définir une file à double extrémité (la deque) de 5 éléments,*
2. *initialiser la deque par des nombres au hazard (fonction rand())*
3. *afficher la deque*
4. *on y fait entrer successivement 5 nombres au hazard placés en tête, le dernier élément de la file étant chaque fois retiré. la deque est affichée après chaque opération.*

Exécution :

```
Initialisation de la deque : 7 9 3 8 0
Introduction de 2 : 2 7 9 3 8
Introduction de 4 : 4 2 7 9 3
Introduction de 8 : 8 4 2 7 9
Introduction de 3 : 3 8 4 2 7
Introduction de 9 : 9 3 8 4 2
```

▸ **Exercice 5.** *Le conteneur **list***
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. *définir une liste l_philo de philosophes : Platon, Aristote, Descartes, et Kant*
2. *définir une deuxième liste l_math de mathématiciens : Gauss, Laplace, Poincaré, Descartes*
3. *afficher les deux listes triées (sort())*
4. *fusionner les deux listes et stocker le résultat dans une nouvelle liste l_all (vous pouvez utiliser std::back_inserter avec std::merge)*
5. *supprimer les répétitions dans la liste l_all*
6. *inverser l'ordre de la liste l_all puis l'afficher*

Exécution :

```
Liste l_philo : Platon Aristote Descarte Kant
Liste l_math : Gauss Laplace Poincaré Descarte

Liste l_philo triée: Aristote Descarte Kant Platon
Liste l_math triée: Descarte Gauss Laplace Poincaré

Liste l_all : Aristote Descarte Descarte Gauss Kant Laplace Platon Poincaré

Liste l_all sans répétitions: Aristote Descarte Gauss Kant Laplace Platon Poincaré

Liste l_all inversée:  Poincaré Platon Laplace Kant Gauss Descarte Aristote
```

▸ **Exercice 6.** *Le conteneur **map***

Alors que les conteneurs de séquence sont conçus pour des accès séquentiels et aléatoires à leurs éléments via l'index ou un itérateur, les conteneurs associatifs sont conçus pour un accès aléatoire rapide aux éléments à l'aide de clés. La bibliothèque C++ standard fournit quatre conteneurs associatifs : map, multimap, set et multiset.

1. *Créer une table associative (map), qui stocke l'âge de quelques étudiants identifiés par leurs prénoms.*
2. *Parcourir la table (en utilisant les itérateurs) pour afficher le prénom et l'âge de chaque étudiant.*
3. *Utiliser la méthode find() pour vérifier si l'âge de "Marie" est renseigné ou pas. Si'il est renseigné, on l'affiche, sinon on affiche un message d'erreur.*

Indication : Chaque élément d'une map est en réalité constitué d'une clé et d'une valeur. Les itérateurs pointent en réalité sur des pair. Ce sont des objets avec deux attributs publics appelés first (permet de designer la clé) et second (permet de désigner la valeur correspondante)

Exécution :

```
La liste des étudiants et leurs âges :

Ali a 25 ans.
Jean a 22 ans.
Pierre a 20 ans.
Sara a 19 ans.
Sofia a 18 ans.

L'âge de Marie n'existe pas!
```

▸ **Exercice 7.** *Les **foncteurs** et un vector de char*

Rappelons qu'un foncteur est un objet d'une classe pour laquelle l'opération d'appel de fonctions a été surchargée. Autrement dit, si X est une classe de foncteurs, alors X() désigne un objet de cette classe, donc un foncteur.

1. *Créer une classe de foncteurs **Majuscule** qui permet de mettre en Majuscule une lettre.*

   Indication : La classe doit surcharger la fonction operator(), pour pouvoir désigner un objet M de cette classe en utilisant M().

2. *Utiliser le foncteur pour convertir les lettres d'un tableau de char.*

▸ **Exercice 8.** *Les **foncteurs** et list*

1. *Créer une classe de foncteurs **AjoutSiPair** qui ajoute 10 si le nombre donné en paramètre est un nombre pair.*

2. *Ajouter 10 à tous les nombres pairs d'une liste initialiser avec des valeurs en utilisant le foncteur.*