TP N° 7
Bibliothèque de conteneurs STL

Objectifs
– Se familiariser avec divers types de conteneurs présents dans la STL.

Exercices d'introduction

1. Copier les caractères en utilisant copy
2. Transformer un tableau de caractère en minuscule à un tableau de caractères en majuscules
3. Utilisation du template vector et faire un tri de nombre entiers

0 – Vector
Ecrire un petit programme qui manipule le conteneur Vector pour gérer un vecteur d'entiers

1 – Pile, File et Veteur

Ecrire deux classes génériques *Pile* et *File* qui permettent de créer et de manipuler respectivement des piles et des files d'éléments de type quelconque.

Une pile se manipule selon le principe "dernier arrivé, premier servi" (Last In First Out), à savoir que le sommet de la pile (le seul élément auquel on ait accès) est le dernier élément rentré. Empiler consiste à ajouter un élément en fin de pile (lequel deviendra le nouveau sommet). Dépiler consiste à supprimer le sommet (qui est remplacé par l'ex-avant-dernier élément).

Une file se manipule selon le principe "premier arrivé, premier servi" (First In First Out), à savoir que le sommet de la file (le seul élément auquel on ait accès) est le premier élément rentré. Enfiler consiste à ajouter un élément en fin de file (le sommet est donc inchangé). Défiler consiste à supprimer le sommet (qui est remplacé par l'ex-deuxième élément).

Ecrire un programme principal permettant de créer, de remplir et de vider des piles et des files d'éléments divers (entiers, réels...).

2– Le coursier au sac dynamique
Nous allons reprendre l'exercice du coursier et modifier sa classe Coursier, afin de lui conférer un sac de volume toujours limité, mais de quantité cette fois illimitée. Pour ce faire, nous allons utiliser le conteneur *vector*. Le sac demeure donc un tableau de courriers, pour lequel on doit pouvoir être renseigné à tout moment sur la quantité et le volume de courriers qu'il contient. A la création, le sac est vide.

Effectuer les modifications demandées sur la classe *Coursier*. Y a-t-il beaucoup de modifications à faire sur cette classe et sur le programme principal ? Aurait-on pu passer par le conteneur *list*, plutôt que *vector* ? Pourquoi ?

3 – Le problème de Josephus avec une file standard

Nous allons reprendre l'exercice du problème de Josephus vue la semaine dernière afin de passer, non plus par la classe file que nous avons créée la dernière fois, mais par le conteneur *queue* standard.

Modélisez et résolvez le problème de Josephus pour plusieurs valeurs, mais cette fois au moyen d'un objet de type *queue<int>*.

4 – Contacts et repertoire

Nous souhaitons implémenter en C++ un annuaire téléphonique. Pour ce faire, nous allons créer une classe *Contact* qui contiendra pour chaque personne ses numéros de domicile, de portable, et de travail.

Les numéros seront stockés au moyen du conteneur *map*, qui à chaque type de numéro (“Domicile”, “Travail” ou “Portable”) associera le numéro de téléphone correspondant. On doit pouvoir créer des contacts avec et sans coordonnées, ainsi que des méthodes d'ajout et de modification de numéros. On doit également pouvoir comparer deux contacts sur leur nom et leur prénom : l'un précède-t-il l'autre dans l'ordre alphabétique ?

5 – Polynôme

On définit un polynôme p comme un vecteur de doubles. Chaque valeur du vecteur représente le coefficient d’un monôme du polynôme et le degré son indice dans le vecteur

1. Implémenter la classe Polynôme
2. Implémenter les méthodes ci-dessous de la classe Polynôme
3. Implémenter les fonctions externes suivantes
4.

-ostream& **operator**<<(ostream& sortie, **const** Polynome& polynome);
-Polynome **operator***(**double** x, Polynome **const** & p);
-void affiche_res(**const** Polynome& p, **const** Polynome& q, **const** Polynome& r, **const** Polynome& s) ;