TP Conteneurs

Les tableaux « vector » de la STL

Dans cet exercice, on va réaliser un programme vecteur.cpp qui permet de stocker des std::string et manipuler ceux-ci de manière simple.

Question 1. Vous devez déclarer un vecteur de nom monVecteur stockant des std::string.

Question 2. Ajoutez 5 std::string dans le vecteur qui vaudront respectivement "bonjour", "comment", "allez", "vous", " ?".

Question 3. Affichez la taille de votre vecteur. Afficher sa capacité (capacity). Quel est la différence avec sa taille ?

Question 4. Afficher le contenu du vecteur en utilisant la notation indexée de tableau en accédant à la valeur avec une syntaxe du type : monVecteur[k].

Question 5. Afficher le contenu du vecteur en utilisant les itérateurs (iterator) sur votre vecteur.

Question 6. Réaliser un échange entre le contenu de la case d'indice 1 et le contenu de la case d'indice 3 de votre vecteur (vérifiez votre résultat en affichant le vecteur). Notez l'existence de std::swap.

Question 7. Trier le vecteur en utilisant un algorithme de la STL (inclure l'en-tête algorithm). L'ordre de tri par défaut est celui de la comparaison sur des std::string. Afficher le résultat obtenu.

Question 8. Créer une fonction *affiche( )* qui affiche le contenu du vecteur passé en paramètre. Chaque élément sera espacé d'un 'espace' à l'affichage. Notez qu'ici, on passera le vecteur sous forme de référence constante car il n'a pas à être modifié, ni copié.

Question 9. Créer une fonction *concatene( )* qui concatène l'ensemble des éléments du vecteur dans une seule variable de type std::string. Chaque élément sera espacé d'un 'espace' dans la std::string.
Réfléchir au prototype de votre fonction, sous quelle forme passez-vous le paramètre d'entrée, sous quelle forme retournez-vous le std::string ?

Question 10. Insérer la valeur "a tous" après le premier élément dans votre vecteur. Vérifier votre résultat.

Conseils : utilisez le site : http://www.cplusplus.com/reference/ en priorité !

Remarque : vous pourrez insérer la ligne *using namespace std ;* pour éviter d'avoir à préciser à chaque fois *std ::string*

Notions d' itérateurs

Les **itérateurs** (*iterator*) sont une généralisation des **pointeurs** : ce sont des **objets qui pointent sur d'autres objets**.
Comme son nom l'indique, les itérateurs sont utilisés pour **parcourir une série d'objets** de telle façon que, si on incrémente l'itérateur, il désignera l'objet suivant de la série.

```
int main()
{
    vector<int> v2(4, 100); // un vecteur de 4 entiers initialisés avec la valeur 100
    cout << "Le vecteur v2 contient " << v2.size() << " entiers : ";
    // utilisation d'un itérateur pour parcourir le vecteur v2
    for (vector<int>::iterator it = v2.begin(); it != v2.end(); ++it)
    {
        cout << ' ' << *it;
        cout << '\n';
    }
    return 0;
}
```

Vous pouvez remarquer la pré-incrémentation (++it), cela permet de sécuriser l'exécution du code et être sûr de pointer sur un objet.

On peut effectuer un parcours du début à la fin du conteneur ou alors partir de la fin ; dans ce cas on parle de « reverse operator ».

Itérateur inverse :

Parcours du conteneur de la fin au début

```
// parcours avec un itérateur en inverse
for ( std::vector<int>::reverse_iterator i = v.rbegin(); i != v.rend(); ++i )
{
    cout << *i << '\t';
}
```