

Sujet de TD n°06

séances n°12 & 13

Notions abordées

-
- ✓ std::string, std::vector
 - ✓ std::sort, std::set, std::map
 - ✓ range based for
-

Sur ce TD on vous demande de coder plusieurs fonctions. Notez qu'en TP, vous devez aussi écrire, au fur et à mesure, le programme principal qui teste toutes ces fonctions. C'est toujours une bonne habitude de tester très souvent, pour ne pas avoir un gros morceau de code qui ne fonctionne pas et dans lequel il devient difficile d'identifier les erreurs.

Exercice 1 : Eléments uniques

1. Écrivez une fonction qui demande à l'utilisateur de rentrer une liste d'entiers, un par ligne, et de rentrer une ligne vide pour terminer. Cette fonction doit renvoyer le std::vector<int> qui contient tous les entiers saisis en ordre. Notez que std::getline(std::cin, s) est une fonction pour lire une ligne de stdin et la stocker dans s et std::stoi(s) est une fonction pour convertir un string vers un entier.
2. Écrivez une fonction qui reçoit un const std::vector<int>& et affiche ses entiers, un par ligne. N'utilisez pas d'indice, mais un *range based for* (for avec : et pas de ;).
3. Écrivez une fonction qui reçoit un const std::vector<int>& et affiche ses entiers, un par ligne, mais cette fois en ordre croissante. Utilisez std::sort(...).
4. Dans un tableau trié, les doublons sont obligatoirement consécutifs. Utilisez cette information pour écrire une fonction qui affiche les éléments en ordre croissant, mais en affichant seulement une fois chaque élément, peu importe le nombre d'occurrences. Utilisez un *range based for* pour cela.
TP : Cherchez comment utiliser les fonctions std::unique et std::erase pour faire la même tache.
5. Le type abstrait de donnée std::set représente un ensemble au sens mathématique du terme : il n'y a pas de doublon et les éléments sont visités en ordre croissant. Utilisez cette structure pour obtenir le même résultat que celui de la question précédente mais sans utiliser la méthode std::sort.

Exercice 2 : Manipulation de std::strings

1. Écrivez une fonction qui remplace toutes les occurrences d'une voyelle minuscule (pas d'accent) par le numéro correspondant : a→1, e→2, i→3, o→4, u→5. Utilisez une boucle *range-based for*.
ex : vichysoise → v3chys43s2
Une possibilité est d'utiliser un *switch* pour traiter les 5 voyelles une par une. Une autre possibilité plus avancée est d'utiliser un std::map.
2. Écrivez une fonction qui remplace une chaîne de caractère s par la chaîne s², obtenue par la concaténation de s avec elle-même.
ex : maison → maisonmaison
3. Écrivez une fonction qui reçoit un string x et renvoie vrai s'il existe une chaîne s tel que x = s².
ex : maisonmaison → vrai
4. Écrivez une fonction qui reçoit un string et renvoie un vector de string en coupant le string

par le caractère espace.

ex : "J'aime le langage C++" $\rightarrow \{"\text{J'aime}", \text{"le"}, \text{"langage"}, \text{"C++"}\}$

Exercice 3 : Arithmétique de taille non bornée

1. Écrivez une fonction qui reçoit un vector d'entiers entre 0 et 9 et renvoie un string avec les chiffres correspondants en ordre inversé.

ex : $\{5, 3, 1, 0, 4\} \rightarrow \text{"40135"}$

2. La fonction précédente établi une correspondance entre ce genre de vector et les entiers positifs. Écrivez une fonction qui augmente d'une unité le numéro correspondant au vector donné, renvoyant un nouveau vector.

ex : $\{9, 9, 1, 3\} \rightarrow \{0, 0, 2, 3\}$ (parce que $3199 + 1 = 3200$)

Cette fonction va utiliser l'addition que vous avez apprise en école primaire. Vous allez parcourir les éléments du vector en ordre en ajoutant une unité. Si la valeur devient 10, on met 0 et pas 10 et on continue en propageant la retenue, sinon, on s'arrête.

3. Écrivez une fonction qui calcule la somme de deux numéros représentés comme deux vector. N'utilisez pas la fonction précédente pour ça, mais l'algorithme qui vous avez appris à l'école primaire. Pensez bien à comment l'algorithme fonctionne et comment le coder.

ex : $\{8, 1, 6, 7\}, \{5, 3, 4, 2\} \rightarrow \{3, 5, 0, 0, 1\}$ (parce que $7618 + 2435 = 10053$)

4. Créer un vector F avec les nombres de Fibonacci : $F[0] = 0$, $F[1] = 1$ et $F[i] = F[i-2] + F[i-1]$. Allez jusqu'à 200. Notez que $F[95]$ dépasse déjà la capacité d'un *unsigned long long int* (64 bits), mais ici avec cette représentation cela ne vous posera aucun problème. Vous pouvez aller bien plus loin !

Par exemple : $F[200] = 280571172992510140037611932413038677189525$