SMP - TP9

Ce TP a pour simple objectif de vous faire utiliser quelques classes de la "Standard Library" de C++. Il n'est constitué que de petits exercices indépendants. Pour pouvoir les faire vous **devez** regarder la documentation associée à chaque classe `https://www.cplusplus.com/reference/`.

1 VECTEUR D'ENTIERS

Créez un code qui permet de faire les actions suivantes :

- définissez d'un vecteur stockant des entiers sans spécifier de taille
- dimensionnez le a une taille de 3
- affectez 3 valeurs
- parcourez et affichez votre vecteur de 3 manières différentes :
  - avec des indices
  - en exécutant une boucle sur une plage de nombres (C++11 *range based loop* `https://en.cppreference.com/w/cpp/language/range-for`)
  - à l'aide d'un itérateur
- ajoutez un nouvel élément en fin de vecteur avec vérifiez que la taille a été modifiée
- copiez le premier vecteur dans d'autres vecteurs en utilisant `copy` et/ou `assign`

2 VECTEUR DE CHAINES DE CARACTÈRES

Créez un code qui permet de faire les actions suivantes :

- Déclarez un vecteur stockant des std::string.
- Ajoutez cinq `std::string` dans ce vecteur. Ces `std::string` contiendront respectivement "Bonjour", "comment", "allez", "vous", "?".
- Affichez la taille de votre vecteur

— Affichez sa capacité (capacity). Quelle est la différence avec sa taille ? (ajoutez des éléments dans le vecteur et observez les différences entre la taille et la capacité).
— Affichez le contenu du vecteur sur une même ligne (chaque mot étant séparé par un espace) avec les mêmes trois méthodes différentes que précédemment
— Réalisez un échange entre le contenu de la case d'indice 1 avec le contenu de la case d'indice 3 de votre vecteur (vérifiez votre résultat en affichant le vecteur modifié). Utilisez `swap`.
— Insérez la valeur "a tous" après le premier élément dans votre vecteur. Vérifiez votre résultat.
— Changez le point d'interrogation final par "???". Vérifiez votre résultat.
— Affichez le contenu du vecteur en séparant chaque chaine par une virgule.
— Triez le vecteur en utilisant un algorithme de la STL. L'ordre de tri par défaut est celui de la comparaison alphabétique sur des std::string. Affichez le résultat obtenu.
— Créez une fonction affiche qui affiche le contenu du vecteur passé en paramètre. Notez qu'ici, on prendra soin de passer le vecteur sous forme de référence constante car il n'a pas à être modifié ni copié (`std::vector<std::string> const&`).
— Créez 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` de sortie.
— Créez une fonction `ajoute_virgule` qui ajoute une virgule derrière chaque mot contenu dans le `std::vector`. Cette fois, la variable de type `std::vector` passée en paramètre de la fonction doit être modifiée. Utilisez `for_each` et soit une fonction, soit un foncteur.

3 Les listes

— Créez une liste de huit entiers.
— Supprimez le troisième élément.
— Affichez à nouveau votre liste.

Remarque : vous **devez** utiliser des itérateurs.

4 Les maps

— Créez un code qui contient une `std::map` qui, pour chaque année, associe une liste d'évènements (sous la forme d'une std::list). Chaque évènement étant stocké dans une std::string. La map sera ordonnée dans l'ordre chronologique, mais les évènements associés à une date donnée seront ordonnés suivant leur ordre d'entrée dans la structure.
— Construisez une fonction `ajoute_evenement` qui ajoute un événement (date, intitulé) dans votre structure que vous passerez en paramètre.
  Remarque :
  — Il faut séparer les cas où l'on ajoute un événement dans une liste déjà existante, du cas où il faut ajouter une nouvelle liste dans la map.
  — Il est possible d'ajouter une entrée dans une map sous différentes formes : `M[clé]=valeur` ou `M.insert(std::make_pair(clé,valeur))`

— Faites une fonction qui affiche l'ensemble des dates et des événements associés.
— Faites des jeux d'essais en choisissant par exemple des dates historiques et des descriptions associées. Pour vous inspirer, vous pouvez utiliser les données de `dates.csv` fourni.

5 LECTURE/ÉCRITURE DE FICHIER CSV

Écrire une classe csv qui permet de lire et écrire un fichier (date ; descriptions) au format CSV (*Comma-Separated Values* `https://fr.wikipedia.org/wiki/Comma-separated_values`). La lecture stockera le résultat dans une `map` comme définit précédemment. L'écriture sauvera la liste des dates/descriptions que vous aviez créé dans la `map` dans un fichier .csv. Vous compléterez votre classe `CSV` des fonctions `affiche` et `ajoute_evenement` comme définies précédemment.