TP POO C++ : Les conteneurs STL

TP POO C++ : Les conteneurs STL

Les objectifs de ce TP sont de comprendre et mettre en oeuvre les conteneurs STL en C++. Les exercices sont inspirés du site www.france-ioi.org.

Les critères d'évaluation de ce TP sont :

- le minimum attendu : on doit pouvoir fabriquer un exécutable et le lancer !
- le respect des noms de fichiers
- le choix des conteneurs
- le nommage des variables ainsi que leurs types, l'utilisation de constantes
- le code est fonctionnel
- la simplicité du code

Voyageur de commerce

Le fameux onguent de l'université étant une très grande découverte scientifique, vous décidez d'aider les chercheurs qui le fabriquent. Ceci vous amènera à voyager en compagnie de marchands afin de collecter les différents ingrédients nécessaires à la fabrication de l'onguent.

Votre voyage au sein de la caravane des marchands vous a amené de ville en ville, et les ingrédients nécessaires à la fabrication de l'onguent sont achetés un par un.

Étude de marché

Afin de partir dans un long voyage, à la recherche de produits exotiques, les marchands prévoient toujours d'emmener avec eux des produits locaux afin de les vendre au cours du trajet. Pour décider quels produits emmener, ils ont fait une petite étude de marché auprès de la population, en demandant à chaque personne d'indiquer LE produit qu'elle serait prête à acheter (celui qu'elle préfère donc). Il y a 10 produits au total.

☞ Ce que doit faire votre programme :

On vous donne le numéro du produit préféré par différentes personnes. Écrivez un programme qui indique pour chaque numéro de produit, le nombre de personnes dont c'est le produit préféré.

Le premier entier à lire est le nombre de personnes `nbPersonnes` (`nbPersonnes` $\leq$ `1000`) ayant exprimé leur souhait. On lit ensuite `nbPersonnes` entiers : les numéros des produits préférés des différentes personnes. Les produits sont numérotés de `0` à `nbProduits - 1`.

Vous devez afficher `nbProduits` entiers : pour chaque produit dans l'ordre de leur numéro, affichez le nombre de personnes qui le préfèrent.

☞ Exemples :

```
$ ./etude-marche
10
0
2
2
1
2
2
```

```
0
2
3
0

3
1
5
1
```

Question 1. Choisir un conteneur STL pour le comptage des produits. Justifier votre choix.

Question 2. Écrire le programme `etude-marche.cpp`. Tester et valider.

Grand inventaire

Vous êtes dans la boutique du plus grand marchand de la ville, à la recherche d'un certain nombre d'ingrédients. Malheureusement pour vous, c'est la période du grand inventaire et cela peut durer très longtemps ! Vous décidez de les aider. À partir du livre de comptes, sur lequel sont indiqués toutes les ventes et achats de chaque produit, vous allez pouvoir rapidement vérifier si les quantités restantes dans les étalages sont bien les bonnes et s'il n'y a pas eu de vols. Il y a 10 produits au total.

☞ Ce que doit faire votre programme :

Un livre de comptes décrit les achats et ventes successives de produits numérotés de 1 à 10. Le livre décrit les opérations depuis une situation où le stock de chacun des produits était de zéro.

Chaque ligne du livre de comptes décrit l'achat (augmentation du stock) ou la vente (réduction du stock) d'une certaine quantité de l'un des 10 produits.

Votre objectif est de déterminer pour chaque produit, la quantité restant dans le stock à l'issue de l'ensemble de ces achats et ventes.

Le premier entier à lire est `nbOperations` : le nombre d'opérations décrites dans le livre de comptes.

Suivent ensuite `nbOperations` paires d'entiers, où le premier entier de chaque paire est le numéro de l'ingrédient concerné par l'opération, et le deuxième est la quantité. Si la quantité est négative, l'opération est une vente, et si elle est positive, l'opération est un achat du produit indiqué.

Vous devez afficher la quantité restante pour chacun des produits dans l'ordre de leur numéro, une fois l'ensemble des opérations décrites dans le livre effectuées.

☞ Exemples :

```
$ ./grand-inventaire
5
1
100
2
50
1
-50
3
20
2
-10
```

```
50
40
20
0
0
0
0
0
0
0
```

On vous demande de décomposer le programme en trois fonctions :

- `saisirOperations` : pour effectuer la saisie de la liste des `nbOperations` opérations
- `calculerStock` : pour mettre à jour le stock en fonction de la liste des opérations
- `afficherStock` : pour afficher la quantité restante pour chacun des produits dans l'ordre de leur numéro

☞ On privilégiera des passages par références. Faites bien attention de préciser des références constantes à chaque fois que cela sera nécessaire.

✍ *Il est conseillé ici d'utiliser un type structuré pour modéliser une opération :*

```cpp
struct Operation
{
    int numero;
    int quantite;
};

// Exemple : saisie d'une opération
Operation operation;
cin >> operation.numero >> operation.quantite;

// Vous pouvez aussi surcharger l'opérateur >> pour le type Operation
```

Exemple de type structuré

Question 3. Choisir un conteneur STL pour le stock des produits. Justifier votre choix.

Question 4. Choisir un conteneur STL pour la liste des opérations. Justifier votre choix.

☞ Les conteneurs seront déclarés dans le `main` et passés aux différentes fonctions lorsque c'est nécessaire.

Question 5. Écrire les trois fonctions demandées.

Question 6. Écrire le programme `grand-inventaire.cpp`. Tester et valider.

Jour de marché

Lorsqu'on organise un marché, certains emplacements sont beaucoup plus intéressants que d'autres. Afin d'éviter les tentations de « pots de vin », la ville dans laquelle vous venez d'arriver a décidé qu'on organiserait à chaque fois un tirage au sort, chacun ayant sa chance !

Chaque marchand met donc son nom dans un grand panier, et des tirages sont effectués. On tire d'abord le nom du marchand qui aura le premier emplacement, puis celui qui aura le second, et ainsi de suite. On décide alors d'afficher par ordre alphabétique, les noms des marchands avec le numéro de leur emplacement.

☞ Ce que doit faire votre programme :

Votre programme devra lire le nombre d'emplacements `nbEmplacements` (au maximum 1000), puis pour chaque emplacement à partir de `0`, le nom du marchand à qui est attribué l'emplacement.

Ensuite, votre programme devra afficher la liste par ordre alphabétique des emplacements attribués : le nom du marchand et le numéro de l'emplacement qui lui est attribué numéroté à partir de `1`.

☞ Exemples :

```
$ ./jour-marche
4
robert
roger
raymond
alphonse

alphonse 4
raymond 3
robert 1
roger 2
```

Question 7. Choisir un conteneur STL pour le stockage des noms de marchand et des numéros d'emplacement. Justifier votre choix.

Question 8. Écrire le programme `jour-marche.cpp`.

Visite de la mine

L'un des produits nécessaires pour la fabrication de l'onguent magique, un minerai très rare, ne se trouve qu'en un seul endroit sur la planète, au fond de la plus vieille mine existante, jadis exploitée par le peuple nain. Désormais seuls quelques uns d'entre eux sont encore sur place, afin de guider les voyageurs (commerçants et touristes) au sein de ce dédale de cavernes et galeries.

Après avoir engagé un guide, il vous mène jusqu'à l'endroit prévu mais un petit désaccord sur le paiement de ses services le pousse à vous laisser sur place, sans aucune chance de retrouver la sortie. Heureusement votre robot à conservé en mémoire la suite des déplacements qui vous ont amené de l'entrée jusqu'à votre position actuelle, il ne vous reste plus qu'à suivre le chemin inverse !

☞ Ce que doit faire votre programme :

Il existe 5 types de déplacements, représentés par 5 entiers différents : aller à gauche (1), aller à droite (2), aller tout droit (3), monter (4) et descendre (5).

✍ *Il est conseillé ici d'utiliser un type énuméré pour le déplacement :*

```
typedef enum
{
    AUCUN = 0,
    GAUCHE = 1,
    DROITE,
    DROIT,
```

```
    HAUT,
    BAS
} Deplacement;

// Exemple :
Deplacement deplacement;

// Un déplacement à gauche
deplacement = GAUCHE;

// Pour réaliser une saisie d'un Deplacement avec cin
std::istream& operator>>(std::istream& is, Deplacement& d)
{
    int tmp ;
    if(is >> tmp)
        d = static_cast<Deplacement>(tmp);
    return is ;
}
```

Type enum

Vous devez assurer la saisie de chaque déplacement. La saisie se terminera lorsque le programme lira la valeur 0. Ensuite, vous devez afficher la suite des déplacements à faire pour aller de votre position actuelle à la sortie.

☞ Exemple :

```
$ ./visite-mine
1
3
2
4
4
5
0

4
5
5
1
3
2
```

Question 9. Choisir un conteneur STL pour le stockage des déplacements. Justifier votre choix.

Question 10. Écrire le programme dans le fichier `visite-mine.cpp`. Tester et valider.

Bonus : Étude de marché II

Afin de partir dans un long voyage, à la recherche de produits exotiques, les marchands prévoient toujours d'emmener avec eux des produits locaux afin de les vendre au cours du trajet. Pour décider quels produits emmener, ils ont fait une petite étude de marché auprès de la population, en demandant à chaque personne d'indiquer LE produit qu'elle serait prête à acheter (celui qu'elle préfère donc). Il y a 10 produits au total.

☞ Ce que doit faire votre programme :

On vous donne le numéro du produit préféré par différentes personnes. Écrivez un programme qui indique pour chaque numéro de produit, le nombre de personnes dont c'est le produit préféré.

Le premier entier à lire est le nombre de personnes `nbPersonnes` (`nbPersonnes` $\leq$ 1000) ayant exprimé leur souhait. On lit ensuite `nbPersonnes` entiers : les numéros des produits préférés des différentes personnes. Les produits sont numérotés de 0 à `nbProduits - 1`.

Vous devez afficher `nbProduits` entiers : pour chaque produit **dans l'ordre de leur préférence**, affichez le nombre de personnes qui le préfèrent.

☞ Exemples :

```
$ ./etude-marche
10
0
2
2
1
2
2
0
2
3
0

produit n°2 : 5
produit n°0 : 3
produit n°1 : 1
produit n°3 : 1
```

✍ *On désire maintenant utiliser un type structuré pour modéliser un produit :*

```
struct Produit
{
    int numero;
    int popularite;
    Produit(int numero) : numero(numero), popularite(0) {}
};
```

Le type Produit

La différence entre une classe et une structure en C ++ est que les structures ont des membres publics par défaut et que les classes ont des membres privés par défaut.

✍ *On choisit un conteneur de type* ***vector*** *pour le stockage des produits :*

```
// Exemple : un vector de 10 produits

const int nbProduits = 10;
vector<Produit> produitsPreferes;

produitsPreferes.reserve(nbProduits); // Reservation mémoire pour 10 produits
for (int i = 0; i < nbProduits; ++i)
{
```

```
    produitsPreferes.push_back(Produit(i));
}
```

un vector de 10 Produit

La STL fournit une fonction `sort()` qui permet de trier un `vector` (cf. http://www.cplusplus.com/reference/algorithm/sort/).

Question 11. Pour utiliser la fonction `sort()` sur le `vector`, quel opérateur faudra-t-il surcharger pour trier nos `Produit` par ordre croissant ? par ordre décroissant ?

Question 12. Écrire le programme `etude-marche-v2.cpp`. Tester et valider.