Niveau 2

TP
S4. Développement logiciel
S4.6. Programmation orientée objet
Programmation générique :
structure de la STL, conteneurs et itérateurs

Concessionnaire STL

TABLE DES MATIÈRES

1 - PRÉSENTATION

Le but de ce TP est la mise en œuvre des conteneurs fournis par la STL C++.

Monsieur Kadératé, concessionnaire d'automobile d'origines japonaises, veut gérer informatiquement son parc de véhicules. Il fait appel à vous pour écrire le programme permettant de fournir les caractéristiques de chacun d'entre eux. Yamamoto ne peut pas vous donner le nombre de véhicules de son stock, ce dernier étant très fluctuant. Votre programme doit tenir compte de cet impératif.

Un tableau dynamique semblerait alors être la solution mais, en fonction des ventes et des nouveaux arrivages des véhicules, ce tableau ne serait pas toujours dimensionné correctement, nous obligeant à en créer d'autres mieux adaptés.

Le nom de la concession de M. Kadératé, Concession STL, vous donne l'idée d'utiliser des outils informatiques pour développer une solution élégante à ce problème : les conteneurs.

2 - TRAVAIL À RÉALISER

1) Créez un nouveau projet C++ console, nommé `VecteurVehicule`.

2) Codez la classe `TVehiculeTerrestre` proposée, en l'adaptant de façon à ce qu'elle soit compatible avec l'utilisation des conteneurs de la STL. Codez alors le contenu de ces nouvelles méthodes.

3) Trouvez un moyen d'afficher simultanément tous les attributs d'un objet instance de `TVehiculeTerrestre` ainsi que l'adresse de cet objet. Codez votre solution.

4) Validez votre travail en créant plusieurs objets, instances de `TVehiculeTerrestre`, en utilisant tous les constructeurs disponibles puis affectez un véhicule à un autre. Vous choisirez les noms de ces objets judicieusement. L'affichage de chaque objet sera alors codé.

5) Créez un vecteur de la STL, nommé `oVecteurVT`, qui devra contenir des `TVehiculeTerrestre`. Remplissez ce vecteur avec les divers objets créés précédemment.

6) En utilisant la première méthode d'accès à un vecteur proposée dans le cours, parcourez et affichez chaque élément du `oVecteurVT`.

7) Reprenez la question précédente en utilisant cette fois la seconde méthode.

8) Testez les méthodes `back()`, `begin()`, `end()` et `front()`. Quelles sont les différences notables ? À quoi servent-elles concrètement ?

9) Testez la méthode `insert()`.

10) Testez les méthodes `pop_back()` et `erase()`. Comparez leur implication sur le vecteur. Réalisez le code permettant de vider le vecteur complètement.

3 - BONUS

11) Proposez une interface graphique permettant la saisie d'un véhicule, l'ajout dans le vecteur puis l'affichage complet du son contenu.

4 - ANNEXES

TVehiculeTerrestre
-VitesseActuelle: unsigned int -Puissance: unsigned int -Poids: unsigned int -VitesseMaxi: unsigned int
<<create>>+TVehiculeTerrestre(Puissance: unsigned int, Poids: unsigned int, VitesseMaxi: unsigned int) +get_VitesseActuelle(): unsigned int +get_Puissance(): unsigned int +get_Poids(): unsigned int +get_VitesseMaxi(): unsigned int +set_VitesseActuelle(VitesseActuelle: unsigned int): void +set_Puissance(Puissance: unsigned int): void +set_Poids(Poids: unsigned int): void +set_VitesseMaxi(VitesseMaxi: unsigned int): void