TD/TP 4 C++

Polytech' Paris-Sud
Et4 info
Aleksandr Setkov et Marc Evrard

19 janvier 2015

1 Templates

Il faut reprendre la classe `Vector` que vous aviez implémentée en premier cours.

1. Modifier la classe `Vector` pour obtenir une version générique de type `template` qui peut travailler avec différents types `T` (comme `int, short, double etc .`). Changer l'implémentation de toutes les méthodes. Tester la classe dans le main.
2. Changer les méthodes d'addition, de soustraction, de multiplication et de comparaison, pour travailler avec deux vecteurs de types différents (`T` et `V` respectivement). Conseil : dans la classe `T`, déclarer la deuxième classe `template V` comme une classe `friend`.
3. Reprendre la classe `Complex` et la modifier pour pouvoir travailler avec la classe `Vector`.
4. Spécialiser le constructeur et les méthodes de comparaison de la classe `Vector` pour travailler avec des données de type `std::string`.

2 Exceptions

1. Dans la classe `Vector`, modifier les méthodes pour générer une exception quand on accède à un élément hors des limites du vecteur. Traiter cette exception dans le main.
2. Dans la classe `Vector`, créer une nouvelle classe d'exception `OutOfRange` qui hérite de la classe de base des exceptions `std::exception`. Redéfinir la méthode virtuelle `what()` qui retourne le texte de l'exception.

3 Iterators

1. Dans la classe `Vector`, implémenter une nouvel classe `Iter` pour naviguer dans le vecteur.

4 Travail sur des fichiers textes

Il faut reprendre les classes `Personne, Etudiant, Employe, Manager` que vous aviez implémentées lors d'un des derniers cours.

1. Créer la classe `FilePersonne` pour lire et écrire des données dans des fichiers textes. Implémenter un constructeur, un destructeur et les méthodes pour ouvrir et fermer un fichier. N'oubliez pas de traiter les situations lorsqu'il est impossible d'ouvrir le fichier.

Nom	Type
fPersFile	`td::f`tream
fileName	`td::`tring

   Le format du fichier est le suivant :

   ```
   Employe Nom Prenom 01/01/1900 Salaire NomEntreprise
   Manager Nom Prenom 01/01/1900 Salaire PrimeAnnuelle GroupeSize
      /*on suppose que GroupeSize = 0*/
   Etudiant Nom Prenom 01/01/1900 NumCarteDEtudiant NomUniversite
      NomSpecialite /*chaque entree correspond a une ligne*/
   Personne Nom Prenom 01/01/1900
   ```

   L'ordre peut-être arbitraire et le nombre total n'est pas limité.

2. Créer une méthode pour lire tout le fichier et stocker les données dans `std::vector<Personne$\ast$>`:

   ```
   void readAllEntries(std::vector<Personne*>& vPersonnes);
   ```

   Pour diviser une ligne en plusieurs parties vous pouvez utiliser

   ```
   p1 = str.find_first_of(delim, p0);
   str.substr(p0, p1 - p0);
   ```

   où p0, p1 – positions dans la ligne str ; delim – délimiteur.

   Traiter la situation lorsque le format du fichier n'est pas correct.

3. Implémenter la méthode de la classe `FilePersonne`

   ```
   bool search(const Personne& pers);
   ```

   qui retourne `true` si la personne est trouvée dans le vecteur, sinon `false`.

4. Créer une méthode pour supprimer une Personne.