TD-TP *LES CONTENEURS « list pair map »*

Objectif : Utiliser la structure « `pair` » et les conteneurs « `list` » et « `map` » **de la bibliothèque standard C++.**

Une liste simple

Sachant que dans la bibliothèque standard C++ :

- La classe list est déclarée comme suit :
```
template <class T, class A = allocator <T> >
class list;
```
- La classe list comporte les déclarations :
```
typedef T value_type;          // équivaut à :
typedef value_type* iterator;  // typedef T* iterator;
```
- La méthode void push_back (const value_type&) ajoute un nouvel élément en fin de liste
- Le type iterator défini dans la classe est utilisé pour itérer sur une liste
- Les méthodes iterator begin() et iterator end() retournent respectivement un pointeur sur le début de la liste et sur l'emplacement qui suit le dernier élément de la liste

Travail à faire

1. Représenter en C++, les caractéristiques de la classe list énoncées ci-dessus
2. En vous inspirant des pages 5, 6, 7 et 8 du support de cours TemplateC++ situé sur eLearn, écrire une procédure listeSimple() dans un fichier main.cpp qui :
   a) Sur le modèle de classe générique list, crée une classe ListeS, liste de string
   b) Crée un objet uneListeS, instance de ListeS
   c) Alimente la liste avec les valeurs "Pantxika", "Yann", "Philippe" et "Patrick"
   d) Crée un iterateur pour accéder aux éléments de uneListeS
   e) Initialise iterateur au premier élément de uneListeS
   f) Fait un parcours avant et complet de la liste et affiche chaque élément

Note : Insérer l'instruction #include <list> dans le main.cpp pour pouvoir utiliser la classe générique list définie dans la bibliothèque standard C++.

3. Quelle est la représentation UML de votre code ?
4. Appeler la procédure listeSimple() depuis la fonction principale main().

Une liste de paires

Sachant que dans la bibliothèque standard C++ :

- La classe pair est déclarée comme suit :
```
template <class T1, class T2>
class pair;
```
- La classe pair comporte les déclarations
```
typedef T1 first_type;
typedef T2 second_type;
```
- Le constructeur pair (const first_type&, const second_type&) permet d'initialiser un objet
- Les attributs first et second représentent la 1ère et 2ème composante de la paire

Travail à faire

5. Représenter en C++, les caractéristiques de pair énoncées ci-dessus
6. Ecrire un exemple qui crée et initialise une paire de chaines de caractères.
7. Ecrire une procédure listePaires() dans le fichier main.cpp qui :
   a) Construit une classe ListeP, liste de paires de string
   b) Crée un objet uneListeP, instance de la ListeP
   c) Alimente la liste avec les valeurs ("Pantxika", "06.01.01.01.01"), ("Yann", "06.02.02.02.02"), ("Philippe", "06.03.03.03.03") et ("Patrick", "06.04.04.04.04")
   d) Crée un iterateur pour accéder aux éléments de uneListeP
   e) Initialise iterateur au premier élément de uneListeP
   f) Fait un parcours avant de la liste et affiche les composants des éléments
8. Quelle est la représentation UML de votre code ?
9. Appeler la procédure listePaires() depuis la fonction principale main().

Un map

Sachant que dans la bibliothèque standard C++ :

- La classe map est déclarée dans le #include <map> comme suit
  template <class Key, class T, class Cmp = less <Key>, class A = allocator <T> >
  class map;
- La classe map comporte les déclarations :
  typedef Key key_type;
  typedef T mapped_type;
  typedef pair <const key_type, mapped_type> value_type;
  typedef value_type* iterator; // type "pointeur sur élément"
- La méthode pair <iterator, bool> insert (value_type& x) insère l'élément x dans le map. Si l'insertion se fait correctement, retourne l'adresse ou est inséré x et true. Sinon retourne l'adresse ou est positionné l'élément de même clef que x et false.
- La méthode iterator find(const key_type& k) retourne un pointeur sur un élément de clé k, ou bien la valeur de end() si la clef k n'existe pas dans le map.

Travail à faire

10. Représenter en C++, les caractéristiques de map énoncées ci-dessus
11. Ecrire une procédure leMap() dans le fichier main.cpp qui :
    a) Définit une classe Annuaire, du type map de string ayant string comme clef
    b) Crée un objet unAnnuaire, instance de Annuaire
    c) Crée resultatInsert une paire de valeurs qui récupère le résultat d'une insertion
    d) Alimente l'annuaire avec ("Pantxika", "06.01.01.01.01"), teste le résultat retourné et affiche le message "Insertion BIEN réalisée" ou bien le message "Insertion MAL réalisée".
    e) Duplique la section de code du c) pour constater que la seconde insertion n'aboutit pas puisque dans un map, la clef est unique.
    f) Insère ("Yann","06.02.02.02.02"), ("Philippe", "06.03.03.03.03") et ("Patrick", "06.04.04.04.04")
    g) Crée un iterateur pour accéder aux éléments de unAnnuaire
    h) Initialise l'iterateur au premier élément de unAnnuaire
    i) Fait un parcours avant du map et affiche le second composant de chaque élément accédé
    j) Cherche (cf. positionne) l'iterateur sur l'élément de unAnnuaire ayant "Philippe" comme clef
    k) Analyse la valeur retournée et affiche le numéro de téléphone, ou bien le message "Philippe" clef inconnue.

Note : Insérer l'instruction #include <map> dans le main.cpp pour pouvoir utiliser la classe générique map définie dans la bibliothèque standard C++.

12. Quelle est la représentation UML de votre code ?
13. Appeler la procédure leMap() depuis la fonction principale main().