université PARIS-SACLAY | IUT D'ORSAY

TP 7

Organisons l'équipe de football et le réseau social

(3 séances entre le 23/10 et 10/11)

Objectifs :

- Produire un code bien construit, bien présenté et qui compile sans erreur.
- Découper un programme existant en sous-programmes et en plusieurs fichiers.
- Manipuler des vecteurs passés en paramètres.
- Distinguer l'usage des paramètres passés par valeur ou par référence.
- Produire des tests automatiques.

Consignes générales pour les exercices sur machine

1. Lire entièrement les questions avant de coder.
2. Préparer un jeu d'essai avant de coder.
3. Adopter la décomposition en « 3 fichiers » : un fichier source principal (commun à tous) et pour chaque groupe de sous-programmes un fichier source et un fichier d'en-tête.
4. Faire des tests automatiques dans le `main` en utilisant le jeu d'essais pour toutes les questions marquées par le symbole **[*]**.

Travail à rendre : À la fin de la 3e séance (pas de temps supplémentaire), vous rendrez les fichiers source (.cpp) et en-tête (.h) du code (dûment commentés) que vous aurez produits. Votre `main` devra inclure le programme principal faisant appel aux sous-programmes définis, ainsi que l'ensemble des tests automatiques que vous avez effectués. Votre rendu sera pris en compte pour l'attribution d'un bonus/malus comptant pour votre note de S-101.

Disponible sur Moodle :

- L'énoncé du sujet
- Un fichier source `foot-corrige.cpp`
- Un fichier d'aide pour présenter les tests

Exercice 1 : Organisons l'équipe de football

Dans cet exercice, nous reprenons l'exercice 1 du TP4 dont la correction vous est donnée dans le fichier `corrige-football.cpp`. Vous pourrez ainsi copier et coller des instructions de ce fichier pour écrire les sous-programmes demandés.

L'objectif est de définir des sous-programmes pour traiter les différentes actions de l'exercice 1 du TP4, de les tester avec des tests automatiques dès que cela est possible (questions marquées par le symbole [*]) et de les utiliser dans le programme principal. Les sous-programmes ne doivent pas gérer la partie interaction avec l'utilisateur. Pour cela veiller à définir des sous-programmes qui ne nécessitent pas de saisies utilisateur (celles-ci se feront toutes dans le `main`).

1. Créer un nouveau projet contenant 3 fichiers : `football.cpp` (qui contiendra la fonction main) et `vecteurs.h` et `vecteurs.cpp` (destinés à contenir les sous-programmes de manipulation de vecteurs qui vous sont demandés).
2. Recopier dans `football.cpp` la déclaration du vecteur `equipe` du fichier `corrige-football.cpp` (ligne 11).
3. En vous inspirant des lignes 27 à 30 du fichier `corrige-football.cpp` créer une procédure `affiche` pour afficher le contenu d'un vecteur de string.
   Utiliser cette procédure pour afficher le contenu du vecteur `equipe` dans votre programme `football.cpp`.
   Effectuez manuellement les tests pour cette question. Mettez vos tests en commentaire à la fin du fichier source contenant la définition du sous-programme.
4. **[*]** En vous inspirant des lignes 52 à 54 du fichier `corrige-football.cpp`, écrire une procédure `normalisation` permettant de modifier le nom d'une personne pour le mettre en majuscules.
5. **[*]** En vous inspirant des lignes 63 à 80, écrire une fonction `recherche` permettant de rechercher une personne dans un vecteur. Quels sont ses paramètres ?
   La fonction doit retourner -1 si la personne n'est pas présente et sinon elle retourne l'indice de la case qui contient la personne. Penser à utiliser le sous-programme `normalisation` dans le sous-programme `recherche`.
6. **[*]** En utilisant la fonction `recherche` écrire une fonction `remplacer` pour remplacer une personne par une autre dans un vecteur. Quels sont ses paramètres ?
   Si la personne est présente, le sous-programme remplacer retourne vrai, faux sinon.

Exercice 2 : Facebook

Dans cet exercice, nous reprenons le TP5 dont la correction vous est donnée. Vous pourrez ainsi copier et coller des instructions de ce fichier pour écrire les sous-programmes demandés.

L'objectif est à nouveau de définir, tester et utiliser des sous-programmes pour traiter les différentes actions de cet exercice. Comme dans l'exercice précédent, veiller à définir des sous-programmes qui ne nécessitent pas de saisies utilisateur.

Vous testerez consciencieusement chaque sous-programme.

1. Créer un nouveau projet contenant 3 fichiers : `mainFacebook.cpp` (qui contiendra la fonction main) et `fonctionsFacebook.h` et `fonctionsFacebook.cpp` (destinés à contenir les sous-programmes de manipulation de vecteurs qui vous sont demandés).
2. Ecrire un sous-programme **`initialisation`** permettant d'initialiser un réseau social à partir d'une constante **`NBMENBRES`** donnant le nombre d'utilisateurs.
3. Ecrire un sous-programme un sous-programme **`affichage`** permettant de l'afficher comme vous l'avez fait dans le TP5.
4. **[*]** Ecrire un sous-programme **`ajoutAmis`** qui ajoute au réseau une relation d'amitié entre deux utilisateurs (qui sont déjà dans le réseau, on ne vérifie pas).
5. **[*]** Ecrire un sous-programme qui retourne le nombre d'amis en communs entre deux personnes.
6. **[*]** Ajouter le sous-programme ci-dessous à votre projet. Préparer un jeu d'essai. Tester le programme et apporter les modifications nécessaires.
   Recommencer la boucle modifications-tests autant de fois qu'il faudra.

```
// la fonction groupeAmis retourne vrai si 3 personnes forment un
groupe d'amis où chacun est amis avec les deux autres et faux sinon.
bool groupeAmis(const vect<vect<bool>> &reseau, int a, int b, int c) {
      bool res = false ;
      if( reseau[a][b]==reseau[b][a]==true) res = true ;
      if( reseau[a][c]==reseau[c][a]==true) res = true ;
      if( reseau[c][b]==reseau[b][c]==true) res = true ;
      return(res) ;
}
```

7. On souhaite enregistrer le nom de chaque utilisateur. Pour cela, on utilise un nouveau vecteur où dans la case d'indice i est enregistré le nom de l'utilisateur i du réseau.
   Mettez à jour votre programme principal (et les sous-programmes que vous jugerez utiles) afin d'utiliser le nom des individus et non pas leur indice dans l'affichage et l'ajout de relation d'amitié. **Attention :** cette modification concerne à la fois l'affichage et la saisie de membres du réseau.
8. **(optionnelle)** Ecrire une fonction **`grandGroupeAmis`** qui prend en paramètre un vecteur2D de booléens représentant un réseau social et un vecteur d'indices représentant un ensemble d'utilisateurs du réseau. La fonction retourne vrai si tous les utilisateurs de l'ensemble sont amis avec tous les autres utilisateurs de l'ensemble, et faux sinon.