R1.01 : Initiation au développement (partie 2)
Feuille TP n° 3

version 2

Algorithmes classiques pour tableaux : Tris

OBJECTIFS PEDAGOGIQUES :

1.- Codage d'algorithmes sous forme modulaire : création et utilisation de sous-programmes et de modules, séparation du code dans fichiers de spécification et d'implémentation
2.- S'exercer à l'écriture progressive de programmes.
3.- Réaliser et consigner le test fonctionnel d'un programme.

OBJECTIF PRATIQUE :

- Regrouper dans une bibliothèque (module) nommée **`bibTableaux`** un ensemble de sous-programmes réalisant des opérations sur des tableaux, d'entiers et d'enregistrements de type Personne,
- Coder ces sous-programmes à partir des algorithmes conçus en TD,
- Programmer le test individuel de chaque sous-programme dans le fichier **`main.cpp`**,
- Passer le test fonctionnel de chaque sous-programme et consigner les résultats des tests dans un fichier de tests.

RESSOURCES A VOTRE DISPOSITION POUR REALISER CE TP :

`ressourcesTP3.zip` : une archive composée de
- De deux fichiers **`.txt`** contenant du code à intégrer dans vos modules (si cela n'est pas déjà fait)
- **`main.cpp`** complet : il contient tous les sous-programmes nécessaires à trier vos sous-programmes de tris ; leurs corps sont commentés (entre 2 symboles /* et */)
- **`feuilleTests_tp3.xls`** : une feuille de tests qui vous permettra de consigner les résultats des tests fonctionnels réalisés sur les sous-programmes développés à l'aide du programme main() fourni

EXERCICES A CODER

3 sous-programmes de la feuille de TD n°4 et un sous-programme vu en cours :
- 2 sous-programmes **`triBulle`** :
  - un sous-programme réalisant le tri de la Bulle croissant d'un tableau d'entiers,
  - un sous-programme, de même nom, réalisant le tri de la bulle, selon un ordre croissant sur le nom, d'un tableau de personnes
- sous-programme **`triParInsertion`** : tri par Insertion croissant d'un tableau d'entiers
- sous-programme **`triParSelectionDePlaces`** : tri par Sélection de places croissant d'un tableau d'entiers

A terminer **avant** la prochaine séance de TP (semaine prochaine).

Chaque sous-programme développé est à tester à l'aide :
- du programme de test fourni dans le fichier main.cpp
- et du fichier excel également fourni, à compléter avec les résultats du test.

PREPARATION AU TRAVAIL

1. A l'aide de l'explorateur de fichiers, dans votre espace de travail de programmation vsCode (dossier **`r101_partie2`**), créer un dossier **`tp3`** pour accueillir les fichiers de cette feuille de TP.
2. Sur eLearn, télécharger l'archive **`ressourcesTP3.zip`**, et décompresser son contenu **dans** le dossier **`tp3`**.
3. Compiler.
4. Copier dans le dossier **`tp3`** les fichiers **`tp2/bibTableaux.h`**, **`tp2/bibTableaux.cpp`**, et, si vous les avez créés, les fichiers **`tp2/personne.h`** et **`tp2/personne.cpp`**
5. Dans la fonction **`main()`**, décommenter la ligne `10 //#include "bibTableaux.h"`
6. Compilez. Si les fichiers importés de votre dossier **`tp2`** sont corrects, la compilation devrait être OK.

MISE EN PLACE

Ajouter le module `Personne`

7. Si vous avez déjà créé le module Personne lors de la séance de TP n°2, passez au point 12.
8. Dans le dossier **`tp3`**, créer 2 fichiers, **`personne.h`** et **`personne.cpp`**, correspondant, respectivement à l'Interface et au Corps du module **`Personne`**.
9. Utiliser le contenu du fichier **`ressourcesPersonne.txt`** pour **constituer** et **lier entre eux** les fichiers Interface et Corps du module **`Personne`**. Penser aux gardes et aux directives d'inclusion.
10. Ajouter dans l'Interface de **`bibTableaux`** la directive d'inclusion du module **`Personne`**
11. Compiler.
12. Supprimer le fichier **`ressourcesPersonne.txt`**

Compléter le module `bibTableaux`

13. Utiliser le contenu du fichier **`ressourcesBibTableaux.txt`** pour **compléter** les fichiers Interface et Corps du module **`bibTableaux`**.
14. Compiler.
15. Supprimer le fichier **`ressourcesBibTableaux.txt`**
16. **Questions :**
    - Combien de sous-programmes **`afficher()`** y a-t-il dans le module **`bibTableaux`** ?
    - Comment s'appelle la situation où plusieurs sous-programmes ont le même nom ?
    - Comment le compilateur peut-il différencier 2 sous-programmes ayant le même nom ?

SOUS-PROGRAMME - `triBulle()` triant un tableau d'entiers

Ajouter le sous-programme au module `bibTableaux`

17. Ajouter la **déclaration** du sous-programme **dans l'Interface** du module et le **corps minimal** du sous-programme **dans le corps** du module. Penser aux gardes et aux directives d'inclusion.
18. Compiler. Corriger la/les erreur(s) avant de passer à l'étape suivante.
19. Coder le corps du sous-programme **de manière incrémentale**, en compilant régulièrement.

Passer le test

20. Avant cela, dans le fichier **`main.cpp`**, analyser le sous-programme de test **`testTriBulle_entiers()`** : vérifier qu'il teste le jeu d'essai de la feuille de tests.
21. Puis passer le test :
    - Dans le fichier **`main.cpp`**, décommenter le contenu du sous-programme **`testTriBulle_entiers()`**,
    - Dans la fonction **`main()`**, appeler le programme **`testTriBulle_entiers()`**
    - Compiler
    - Exécuter la fonction **`main()`,**
    - Noter les valeurs obtenues sur le fichier de test excel fourni,
    - Et comparer les **valeurs obtenues** (fournies par l'exécution du programme) avec les **valeurs attendues** (celles figurant dans la colonne 'valeurs attendues' du fichier excel de tests).

SOUS-PROGRAMME - `triBulle()` triant un tableau de personnes

Ajouter le sous-programme au module `bibTableaux`

22. Ajouter la **déclaration** du sous-programme **dans l'Interface** du module et le **corps minimal** du sous-programme **dans le corps** du module. Penser aux gardes et aux directives d'inclusion.
23. Compiler. Corriger la/les erreur(s) avant de passer à l'étape suivante.
24. Coder le corps du sous-programme **de manière incrémentale**, en compilant régulièrement.

Passer le test

25. Avant cela, dans le fichier **`main.cpp`**, analyser le sous-programme de test **`testTriBulle_personnes()`** : vérifier qu'il teste le jeu d'essai de la feuille de tests
26. Puis passer le test :
    - Dans le fichier **`main.cpp`**, décommenter le contenu du sous-programme **`testTriBulle_personnes()`**,
    - Dans la fonction **`main()`**, appeler le programme **`testTriBulle_personnes()`**
    - Compiler
    - Exécuter la fonction **`main()`,**
    - Noter les valeurs obtenues sur le fichier de test excel fourni,
    - Et comparer les **valeurs obtenues** (fournies par l'exécution du programme) avec les **valeurs attendues** (celles figurant dans la colonne 'valeurs attendues' du fichier excel de tests).
27. **Questions :**
    - Dans le tableau2, **avant** le tri, lister les prénoms des 3 individus nommés Marquesuzaa, dans l'ordre d'apparition
    - Faites de même, une fois le tableau2 trié.
    - Que constatez-vous ?
    - Comment s'appelle cette propriété ?

SOUS-PROGRAMME – `triParInsertion()` triant un tableau d'entiers

Ajouter le sous-programme au module `bibTableaux`

28. Ajouter la **déclaration** du sous-programme **dans l'Interface** du module et le **corps minimal** du sous-programme **dans le corps** du module. Penser aux gardes et aux directives d'inclusion.
29. Compiler. Corriger la/les erreur(s) avant de passer à l'étape suivante.
30. Coder le corps du sous-programme **de manière incrémentale**, en compilant régulièrement.

Passer le test

31. Avant cela, dans le fichier **`main.cpp`**, analyser le sous-programme de test **`testParInsertion_entiers()`** : vérifier qu'il teste le jeu d'essai de la feuille de tests
32. Puis passer le test :
    - Dans le fichier **`main.cpp`**, décommenter le contenu du sous-programme **`testTriParInsertion_entiers()`**,
    - Dans la fonction **`main()`**, appeler le programme **`testTriParInsertion_entiers()`**
    - Compiler
    - Exécuter la fonction **`main()`,**
    - Noter les valeurs obtenues sur le fichier de test excel fourni,
    - Et comparer les **valeurs obtenues** (fournies par l'exécution du programme) avec les **valeurs attendues** (celles figurant dans la colonne 'valeurs attendues' du fichier excel de tests).

COMPTER LE NOMBRE D'ACCES AU TABLEAU DANS LE `triBulle()` triant un tableau d'entiers

Adapter le sous-programme le sous-programme au module `faireMonterLaBulleIci()`

Ajouter une nouvelle version du sous-programme **`faireMonterLaBulleIci`**, permettant de compter et retourner le nombre d'accès au tableau qui sont réalisés lors de l'exécution du sous-programme.

Sa déclaration pourrait être :

```
void faireMonterLaBulleIci (int tab[] , unsigned int bDeb,
                            unsigned int ici, unsigned int& nbAcces);
/* Fait remonter le plus grand élément de la tranche tab(0..ici] à l'indice ici
   par échanges successifs.
   nbAcces précise le nombre d'accès au tableau réalisés dans le sous-programme*/
```

33. Créer son corps, par duplication de la version actuelle du sous-programme, puis modifier le coprs de sorte à comptabiliser tout accès au tableau.
34. Compiler. Corriger la/les erreur(s) avant de passer à l'étape suivante.
35. Ajouter les instructions au sous-programme **`triBulle()`** d'un tableau d'entiers de sorte qu'il affiche le nombre total d'accès au tableau une fois le tri réalisé.

Passer le test…

36.
- Exécuter le dans la fonction **`main()`** le sous-programme de test adéquat,
- Pour chaque tableau trié, le triBulle affichera le nombre d'accès au tableau durant le tri. Expliquez les différences de valeurs obtenues en fonction des tableaux fournis.

37. Retrouvez ces valeurs en répondant à la question B de l'exercice n°1 de la feuille de TD n°4.

SOUS-PROGRAMME – `triParSelectionDePlace()` triant un tableau d'entiers

Ajouter le sous-programme au module `bibTableaux`

38. Ajouter la **déclaration** du sous-programme **dans l'Interface** du module et le **corps minimal** du sous-programme **dans le corps** du module. Penser aux gardes et aux directives d'inclusion.
39. Compiler. Corriger la/les erreur(s) avant de passer à l'étape suivante.
40. Coder le corps du sous-programme **de manière incrémentale**, en compilant régulièrement.
41. **Questions :**
    - Quel est le statut du sous-programme **`echanger()`**?
    - A-t-il une place particulière dans le corps du module **`bibTableaux`** ?

Passer le test

42. Avant cela, dans le fichier **`main.cpp`**, analyser le sous-programme de test **`testTriParSelectionDePlace_entiers()`** : vérifier qu'il teste le jeu d'essai de la feuille de tests
43. Puis passer le test :
    - Dans le fichier **`main.cpp`**, décommenter le contenu du sous-programme **`testTriParSelectionDePlace_entiers()`**,
    - Dans la fonction **`main()`**, appeler le programme **`testTriParSelectionDePlace_entiers()`**
    - Compiler
    - Exécuter la fonction **`main()`,**
    - noter les valeurs obtenues sur le fichier de test excel fourni,
    - et comparer les **valeurs obtenues** (fournies par l'exécution du programme) avec les **valeurs attendues** (celles figurant dans la colonne 'valeurs attendues' du fichier excel de tests).

SUPPRIMER L'ARCHIVE `ressourcesTP3.zip`

RAPPELS

Avant de coder :

- Faire un algorithme
- Préparer le jeu d'essai servant à écrire le sous-programme de test
  - Ajouter, dans le fichier de tests excel fourni, un onglet spécifique au sous-programme
  - Créer le tableau avec les valeurs qui seront fournies au sous-programme testé, et les valeurs attendues

Lors du codage : appliquer toutes les recommandations vues dans la première partie de R1.01-Initiation au développement (partie 1).