IFT1169 – TRAVAIL PRATIQUE #3 – 17 mars 2007

« Place aux templates boostées avec des algorithmes sous toutes les sauces! »

Mohamed Lokbani

Équipes: le travail peut-être fait en binôme. Vous ne remettez alors qu'un travail par équipe.
Remise : une seule remise à effectuer par la voie électronique le dimanche **08 avril 2007, 23h59 au plus tard, sans possibilités de prorogation**.
Conseils: n'attendez pas le dernier jour avant la remise pour commencer. Vous n'aurez pas le temps nécessaire pour le faire.

But : ce TP a pour but de vous faire pratiquer les fonctions génériques et l'utilisation de la « STL »..

Exercice 1 : on vous demande d'écrire 4 variantes d'une fonction générique qui a un seul argument, un vecteur de type quelconque. Cette fonction générique retourne l'élément le plus élevé dans ce vecteur.

- Approche directe (2 variantes) : chaque variante va utiliser une des fonctions définies dans le fichier « <algorithm> ». Le corps de la fonction ne va pas dépasser dans ce cas 1 à 2 lignes de code. Une indication : comme première fonction vous pouvez utiliser « sort », on vous laisse chercher la seconde fonction. Voir ce lien pour une liste complète des fonctions définies dans « <algorithm> » : http://www.cplusplus.com/reference/algorithm/

- Approche indirecte (2 variantes): chaque variante va utiliser la bonne vieille méthode, la boucle « for », pour trouver l'élément le plus élevé dans le vecteur. Une indication : comme première solution, vous pouvez utiliser une boucle « for » sur des « int ». On vous laisse chercher la seconde façon de faire tout en utilisant toujours la boucle « for ». Le corps de la fonction ne devrait pas dépasser la dizaine de lignes de code.

Finalement, on vous demande d'écrire un programme permettant de tester les 4 fonctions génériques développées précédemment.

Exercice 2 : soit le fichier « entree.txt » qui contient les entrées suivantes :

Identificateur	Valeur associée
`px`	`nom prénom`
`ex`	`prénom`

L'identificateur est composé de deux lettres :

- la première lettre est soit « p » pour parent ou bien « e » pour enfant.
- la seconde lettre « x » est un nombre. En effet, on peut avoir dans le fichier des entrées associées à plusieurs parents et chaque parent peut avoir lui aussi des entrées associées à plusieurs de ses enfants.

Un parent est identifié par son nom de famille et son prénom. Ce sont des variables du type « string ».
Un enfant n'est identifié que par son prénom. C'est une variable du type « string ».

Une description détaillée des fonctionnalités de la classe « string » avec des exemples est disponible à cette adresse :
http://www.iro.umontreal.ca/~pift1166/cours/ift1166/sessions/A05/Tps/Tp3/char911.cpp

Le fichier « entree.txt » peut contenir par exemple :

« p0 » est le parent numéro « 0 », « p5 » est le parent numéro « 5 »,
« e0 » est l'enfant dont le parent est le « 0 » et « e4 » est l'enfant dont le parent est le numéro « 4 ».

On peut avoir des parents ayant des noms identiques ou prénoms identiques mais pas les deux à la fois :

« p0 » Gabriel Peter « p3 » Gabriel Peter	Deux entrées incorrectes. Parents différents « 0 » et « 3 » mais les noms et prénoms sont identiques.

« p0 » Parker Charlie « p3 » Parker Evan	Deux entrées correctes. Les prénoms sont différents.

Un même parent peut avoir plusieurs enfants ayant des prénoms différents. Les enfants de parents différents peuvent avoir le même prénom :

« e0 » Charlie « e0 » Charlie	Deux entrées incorrectes. Deux enfants d'un même parent « 0 » ayant le même prénom « Charlie ».

« e0 » Charlie « e0 » Nina « e0 » Alain	Deux entrées correctes. Prénoms différents.

« e0 » Charlie « e1 » Charlie	Deux entrées correctes. Parents différents.

Les entrées dans le fichier peuvent être organisées de manière désordonnée. Par exemple:

```
p0 a b
e1 f
e1 e
p1 c d
e0 f
```

Tenant compte du format du fichier en entrée, on vous demande de coder un programme qui doit réaliser ce qui suit :

-1- lire le fichier en entrée et stocker ses données dans le conteneur le mieux approprié.

-2- afficher un menu avec les fonctionnalités suivantes :

-a- afficher en sortie dans un format de votre choix la liste des parents et leurs enfants dans un ordre lexicographique. Lors du tri, pour les parents vous allez tenir compte du nom uniquement. S'il est le même pour deux parents, vous allez tenir compte aussi du prénom. Pour les enfants vous n'allez tenir compte que du prénom. Par exemple, on peut avoir en sortie le format d'affichage ci-dessous :

```
<p> Gabriel Peter
<p> Parker Charlie <e> Alan, Charlie, Nina
<p> Parker Evan
```

-b- calculer la moyenne d'enfants par parents i.e. la somme des enfants par le nombre de parents.

-c- permettre à l'usager de chercher si un parent donné est dans la liste ou pas. L'usager doit fournir au moins le nom (de famille). Si le prénom est aussi fourni, la recherche doit tenir compte de ces deux informations (nom et prénom). Dans le cas contraire la recherche doit se limiter au paramètre nom (de famille). Ainsi vous devez afficher tous les résultats possibles qui correspondent à ce même nom de famille. Citons l'exemple suivant :

```
<p> Parker Charlie
<p> Parker Evan
```

On chercher « Parker », nous allons afficher:

```
<p> Parker Charlie
<p> Parker Evan
```

On cherche « Parker Charlie », nous n'allons afficher que:

```
<p> Parker Charlie
```

-d- permettre à l'usager de chercher si un enfant donné est dans la liste ou pas. L'usager doit fournir obligatoirement le prénom de l'enfant. Si ce prénom existe plus d'une fois dans la liste, vous indiquez le nombre d'enfants dans la liste ayant ce même prénom et vous affichez la liste des parents.

Hypothèses et contraintes :

- pour commencer, vous devez déjà respecter les différentes contraintes de programmation précédemment décrites.
- c'est un travail à faire **SEUL** ! Et seul ne signifie pas deux ni dix. Seul signifie aussi **réfléchir seul**. Plagiat équivaudrait à un 0 pour commencer.
- IFT1169 est un cours avancé en C++, donc votre programme doit être écrit en C++ et pas en C ni en Java! Nous n'autoriserons aucune référence au langage C. Par exemple l'instruction [#include <stdio.h>] fait référence au langage C, donc elle n'est pas autorisée. Utilisez plutôt [#include <iostream>].
- ce travail n'est pas un exercice d'algorithmique, donc pas besoin de compliquer le travail pour rien!
- vous vous assurez de ne pas changer les noms des fichiers. Le format de l'affichage en sortie est libre.

Rapport : le fichier « rapport.pdf » va décrire comment vous avez procédé pour réaliser ce travail (pour le contenu d'un rapport voir aussi la « FAQ » sur la page web du cours). Vous devez prévoir deux sortes de rapport : un guide d'utilisation de votre programme qui sera lu par un utilisateur néophyte et un guide pour programmeur lui expliquant la démarche suivie pour la réalisation de ce travail. Le rapport doit-être aux formats « pdf » ou « ps » (voir là aussi la FAQ sur la page web du cours : comment générer un « pdf »).

Remise : il est important de noter que votre TP sera compilé avec gcc3.2.4. Si par choix vous décidez d'utiliser un autre compilateur, vérifiez que le code que vous avez produit (devant normalement fonctionner correctement chez vous) fonctionne aussi bien sur les ordinateurs de la DESI. Pour avoir la version du compilateur, utilisez la commande "**gcc –v**", qui devra donner le numéro de version "**3.2.4**".
Par ailleurs, assurez-vous de la présence de l'option « pedantic » sur la ligne de compilation. (Cette option n'est pas activée par défaut dans l'utilitaire « devcpp ». Il faudra donc penser à l'activer).
Vous devez d'abord regrouper l'ensemble des fichiers dans un seul fichier « tp3.zip ». Par la suite vous devez faire la remise comme suit :

1. Connectez-vous à « https://remous.iro.umontreal.ca/client » et envoyez le fichier « tp3.zip[1] » via la page web du « tp3 ».

2. Envoyez-nous par email à l'adresse « dift1169@iro.umontreal.ca » avec comme sujet « remise tp#3 », une copie du fichier « tp3.zip ».

Barème : ce TP3 est noté sur 12 points. Le détail pour chaque exercice est comme suit :

Exo1 (5 points)

Compilation, respect des spécifications, codage et commentaires	**1.25**
Rapport	**1.75**
Tests	**2**

[1] Il y a un bug lors de la remise d'un fichier avec l'extension « zip ». J'espère que nous allons trouver la solution d'ici la date de remise. Sinon remettez les fichiers un à un.

Exo2 (7 points)

Compilation, respect des spécifications, codage et commentaires	**2,5**
Rapport	**2**
Tests	**2,5**

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ….

- La non remise électronique (volontaire ou par erreur) est sanctionnée par la note 0.
- Les programmes ne contenant pas d'en-tête, -1 point.
- Un programme qui ne compile pas : 0.
- Un programme qui compile mais ne réalise pas les choses prévues dans la spécification : 0.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.25 et plus.
- Le non respect du nom du fichier entraîne une erreur de compilation donc un des points de la spécification n'a pas été respecté : 0.
- Aberration dans le codage : même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long!

Automatisation de la correction : nous allons utiliser des scriptes pour compiler et exécuter votre programme. Assurez-vous de respecter les noms précédemment mentionnés des fichiers, ainsi que les noms des différentes options.

Ainsi donc, au moment de la remise de votre travail, vous avez déjà une idée approximative de la note que vous allez obtenir.

Des questions à propos de ce TP?

Une seule adresse : dift1169@iro.umontreal.ca

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre email, au moins la chaîne: [IFT1169] et une référence au **tp03**.

Mise à jour

17-03-2006 diffusion de l'énoncé.