IFT1169 – TRAVAIL PRATIQUE #2 – 27 février 2012

« Suite++ »

Mohamed Lokbani

Équipes : le travail est à faire en monôme (**une seule personne**).
Remise : une seule remise est à effectuer par voie électronique **le samedi 24 mars 2012, 23h59 au plus tard, sans possibilités de prorogation**.
Conseils : n'attendez pas le dernier jour avant la remise pour engager votre travail. Vous n'aurez pas le temps nécessaire pour le réaliser.

But : ce TP a pour but de vous faire pratiquer les classes génériques, la manipulation de fichiers et le traitement des exceptions.

Énoncé :

Exercice -1- : nous allons reprendre le travail pratique #2 afin de lui apporter quelques modifications pour tenir compte des notions théoriques apprises dans le cours. Cet exercice est divisé en deux parties :

Question -1- : ajuster le code du tp#2 pour refléter les changements suivants :

- Nous n'allons plus utiliser l'opérateur d'indexation « [ ] ». Il sera remplacé par l'opérateur d'appel de fonction « ( ) ». Ainsi donc « tab[x][y] » devient « tab(x,y) ». Ajuster les différentes méthodes pour refléter ce changement.

  Est-il plus simple de coder l'approche « [ ] » ou l'approche « ( ) »? Pourquoi?

- Nous n'allons plus utiliser l'opérateur de redirection « < » sur la ligne de commandes pour passer les données au programme. Nous allons fournir le nom du fichier en entrée comme argument sur la ligne de commandes. La gestion du fichier passe maintenant entre les mains du langage C++. Quant aux résultats, ils seront affichés par défaut sur la sortie standard sauf si l'utilisateur a précisé l'option « -o fichier_sortie.txt » sur la ligne de commandes. Dans ce cas, les résultats seront écrits directement dans le fichier « fichier_sortie.txt[1] ».

Question -2- : après avoir répondu à la question -1-, compléter votre programme pour qu'il puisse traiter un type quelconque « T ». Donner un exemple dans le cas des types : « int », « string » et « char* ».

Exercice -2- : la durée de vie d'une variable « x » (automatique) est dans le bloc où elle a été créée. Quand une exception est levée, avant que le contrôle passe du bloc « try » à un bloc « catch » correspondant, le langage C++ appelle d'abord le destructeur de tous les objets créés automatiquement depuis le début du bloc « try ». La destruction des objets se fait dans l'ordre inverse de la création. Nous allons examiner cette définition dans deux cas précis.

Question -1- : exception et pointeurs

- Expliquer pourquoi la levée d'une exception dans le fragment de code « 1169Tp3Exo2Q1.cpp », provoque une fuite de mémoire.

- Ajouter les lignes de code nécessaires pour éviter qu'une telle erreur puisse se produire. L'exception est quand même levée au même endroit (le « throw » reste à sa place dans le code).

[1] Le nom « fichier_sortie.txt » est fourni à titre d'exemple. Nous pouvons choisir un autre nom avec l'option « -o ».

Question -2- : le cas du constructeur

Un objet n'est considéré comme étant complètement créé que si le constructeur a terminé l'exécution de toutes ses instructions (dès l'accolade fermante « } »). En cas d'erreur lors de l'exécution d'un constructeur, nous ne pouvons pas retourner un code d'erreur car le constructeur, par conception, ne retourne aucune valeur (et on ne met pas « void »). Les exceptions tombent à pic dans ce cas! Nous allons lever une exception pour signaler l'erreur.
Si une exception est levée dans le constructeur, toute variable (automatique) est détruite comme nous l'avons mentionné au début de cet exercice. Qu'en est-il pour les variables allouées dynamiquement?

- Pourquoi le mécanisme du langage C++ ne va pas faire appel au destructeur associé à l'objet pour détruire ces variables dynamiques créées dans le constructeur?

- Si le mécanisme ne fait pas appel au destructeur, que se passe-t-il, si on ne fait rien, aux éléments alloués dynamiquement avant la levée de l'exception?

- Doit-on corriger la situation, si c'est oui, comment?

Clarifier vos réponses en codant un programme simple « 1169Tp3Exo2Q2.cpp » ayant la classe tableau (monodimensionnel pour simplifier les choses) dont les éléments sont alloués dynamiquement.

Les réponses aux questions doivent figurer dans le rapport. Le programme est utilisé pour étayer les explications.

Fichiers à remettre : vous devez remettre les fichiers nécessaires pour une compilation et une exécution correctes de vos programmes. Vous devez inclure aussi un fichier « makefile » permettant d'automatiser la compilation de vos fichiers. Par ailleurs, vous devez inclure dans votre remise un rapport décrivant le travail effectué. Ce rapport doit porter le nom « tp3H12.pdf ».

Remise : Attention! Vous allez utiliser un nouveau système pour remettre vos travaux!

Il est important de noter que votre TP sera compilé avec « gcc4.6 ». 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 vous donner le numéro de version « **4.6** ».

Par ailleurs, assurez-vous de la présence de l'option « pedantic » sur la ligne de compilation.

Pour éviter d'omettre l'un des fichiers associés à ce travail lors de la remise, nous vous conseillons de regrouper l'ensemble des fichiers dans un seul fichier compressé **tp3h11.zip**.

Il y a quelques règles à respecter pour pouvoir utiliser correctement le nouveau système de remise.

-1- Vous aurez besoin de votre nom d'utilisateur et de votre mot de passe de la DESI.
-2- Vous ne pouvez pas remettre deux fois le même travail dans votre répertoire de remise. S'il en existe déjà, vous devez préalablement l'effacer avant de le remettre de nouveau.
-3- Dès que la remise est terminée, pensez à fermer la fenêtre associée à votre dossier.

Linux (ubuntu)
Menu Fichier > Se connecter au serveur...
Entrer l'adresse du serveur: https://subversion.iro.umontreal.ca/depot/ift1169

Macintosh OSX
Dans le Finder, menu Aller > Se connecter au serveur...
Entrer l'adresse du serveur: https://subversion.iro.umontreal.ca/depot/ift1169

Windows
Utiliser le fichier « RemiseIFT1169.bat » disponible sur la page web du cours. Ce fichier contient en réalité une série de commandes permettant de connecter de manière transparente le premier lecteur réseau disponible au serveur de remise.

Si vous préférez le faire vous-même, vous devez créer un raccourci pour un lecteur réseau (suivre la procédure décrite sur le lien suivant) :
http://windows.microsoft.com/fr-FR/windows-vista/Create-a-shortcut-to-map-a-network-drive
Dans la zone « Emplacement », taper ce chemin d'accès du serveur: https://subversion.iro.umontreal.ca/depot/ift1169

Le répertoire de remise sur le serveur est tp03.

Barème : ce TP3 est noté sur **11 points**. Les points sont répartis comme suit :

Exercice 1	**7** Compilation et respect des spécifications 1 Codage et commentaires, etc. 3 Rapport 2 Tests 2
Exercice 2	**4** Compilation et respect des spécifications Codage et commentaires, etc. Rapport Tests

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 va générer 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!

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

28-02-2012 diffusion de l'énoncé.