# IFT1169 – TRAVAIL PRATIQUE #1 – 30 janvier 2006

## Le chef « C++ » vous laisse deviner quelques unes de ses recettes!

Mohamed Lokbani

**Équipes:** Le travail est à faire en monôme (**une seule personne**).
**Remise :** Deux remises à effectuer : électronique et papier le **mardi 07 février 2006, 22h30 au plus tard, sans possibilités de retard**. Ne seront prises en considération que les copies papiers remises aux démonstrateurs du cours IFT1169 ou bien durant la séance de cours du mardi 07 février 2006.
**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 familiariser avec votre environnement de travail d'une part et d'autre part de manipuler des gadgets de base nécessaires à un programmeur en C++. Nous allons revenir sur l'utilisation des pointeurs, puis nous examinerons par la suite comment extraire et manipuler les différentes options fournies sur une ligne de commande pour finalement vous introduire aux espaces de noms. Le but de ce travail est de vous faire pratiquer aussi la procédure de remise, la rédaction d'un rapport ainsi que le respect des échéances.

**Énoncé :** On vous demande de compléter un programme C++. Pour cela vous devez réaliser ce qui suit :

**-1-** Le programme fourni ne compilera pas pour la simple raison que certaines déclarations sont manquantes. Ces déclarations sont signalées dans le programme par le tag « `Question n` » où « `n` » est le numéro de la question.
Il y a 8 questions à compléter. Pour chacune d'elles, nous vous avons proposé une série d'indices permettant de vous aider dans votre tâche.
Pour vous permettre de répondre à la « `Question 0` », nous avons ajouté **de manière provisoire** l'instruction suivante : « **`int v=10;`** ». Et la réponse à la question « 0 » va être : « `int* var = &v;` ».
Nous avons mis aussi à votre disposition le fichier « **`s01.txt`** » qui correspond à la sortie obtenue suite à l'exécution du programme « `tp1.exe` », généré après compilation de « `tp1.cpp` ».
Certaines questions sont plus difficiles que d'autres et parfois vous devez prendre les questions dans un ordre inversé (de la fin vers le début) pour compléter certaines questions. Pour chaque question, il y a une réponse courte, simple et élégante qui prend en considération la disposition des différentes déclarations dans la méthode « `main` ». C'est cette réponse que nous vous demandons de trouver. D'ailleurs, nous avons mentionné dans plusieurs endroits de la méthode « `main` » quelques réponses parfois trop « évidentes » que l'on ne veut pas lire!
Avant de réaliser la seconde partie de ce travail, nous vous conseillons de compléter d'abord indépendamment et peu importe dans quel ordre, les blocs 1 & 2 de la méthode « `main` ». Attention, en aucun cas, il ne vous est pas permis de retirer des instructions de la méthode « `main` », de changer l'ordre des déclarations, d'en modifier ou d'en rajouter s'il ne vous a pas été clairement mentionné. Globalement, vous ne pouvez que compléter les déclarations demandées associées aux différentes questions et rien d'autre.

**-2-** Compléter par la suite votre programme en y ajoutant l'espace de nom « **`tp1`** » dans lequel il doit figurer les éléments suivants :

- La variable entière « **`v`** » initialisée par défaut avec la valeur « **`10`** ».
- La fonction « **`void aide();`** ». Lors de son appel, elle va afficher en sortie les informations nécessaires permettant d'aider à utiliser correctement ce programme.
- La fonction « **`void init_args(int argc, char** argv);`** ». Elle permet de lire les options fournies sur la ligne de commande, de les analyser et d'agir en conséquence.
- Vous ne devez tenir compte que des 3 options suivantes :
  - [-**`h ou -a`**] : Elles permettent d'obtenir de l'aide sur l'utilisation du programme. Ces options feront appel à la fonction « `aide` ».
  - [-**`v valeur`**] : Cette option va permettre de modifier la valeur par défaut de la variable « `v` ».

Ainsi donc le programme exécutable peut être exécuté sous diverses formes que nous avons résumées en annexe -1-. Vous ne devez tenir compte que de ces appels. Pour vous aider dans cette tâche, nous mettons à votre disposition

plusieurs fichiers correspondants aux différents appels du programme (voir annexe -1-). Vous devez obtenir exactement la même sortie (même les espaces et le nombre de lignes doivent être pris en considération). Par ailleurs, toutes les sorties générées suite à une erreur sur la ligne de commande, doivent être redirigées vers la sortie des erreurs.
Pour les capturer dans un fichier vous pouvez utiliser cette commande « `tp1.exe –a –h > masortie.txt 2>&1` »; ainsi donc les affichages vers la sortie normale (suite à l'utilisation de « `cout` ») et la sortie des erreurs (suite à l'utilisation de « `cerr` ») sont redirigés vers le fichier « `masortie.txt` ».

**-3-** Sachant qu'il ne vous est pas permis d'utiliser la directive « **`using`** », vous devez introduire les ajustements suivants dans la méthode « `main` » :

- Ajouter à l'endroit indiquer dans la méthode « **`main`** », un appel à la méthode « **`init_args`** ».
- Retirer l'instruction « **`int v=10;`** ».
- Faire les ajustements nécessaires aux différentes questions posées en tenant compte que la variable « `v` » est à chercher de l'espace de nom « `tp1` ».

## 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 équivaut 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 non autorisée. Utiliser plutôt [#include <iostream>].
- Ce travail n'est pas un exercice d'algorithmique, donc pas besoin de compliquer le travail pour rien!
- Assurer vous de ne pas changer le nom de fichier, vous devez respecter aussi le format de l'affichage en sortie.
- Les fichiers de sortie ont été générés sous **MinGW/Msys.** Ils sont donc au format DOS étendu. Si vous travaillez sur LINUX, assurez-vous de les convertir au format LINUX en utilisant pour cela la commande **dos2unix**. Pour convertir de LINUX vers le format DOS, utiliser plutôt la commande **unix2dos**. Ces commandes sont disponibles dans un Xterm d'une plateforme LINUX.

**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 (qui normalement fonctionne correctement chez vous) fonctionne bien sur les ordinateurs de la DESI. Pour avoir la version du compilateur, utilisez la commande "**gcc –v**", qui devra donnée le numéro de version "**3.2.4**".

Vous devez remettre 2 fichiers : « tp1.cpp » et « rapport ». Le fichier « tp1.cpp » va contenir votre code alors que le fichier « rapport » va décrire comment vous avez procédé pour réaliser ce travail (pour le contenu d'un rapport voir 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.

La remise comprend **3** (**TROIS**) choses (avant le mardi 07 février 2006 à 22h30) :

1. Envoyez vos fichiers « tp1.cpp » et le rapport par la procédure de remise électronique habituelle (Pour obtenir de l'aide sur cette commande, tapez dans un Xterm : man remise). Respecter les noms des fichiers.

   **remise ift1169 tp1** tp1.cpp rapport

2. Vérifier que la remise s'est effectuée correctement.

   **remise –v ift1169 tp1**

3. Remettez une **copie papier** de votre programme (« tp1.cpp ») ainsi que de votre rapport.

Ne seront prises en considération que les copies papiers remises aux démonstrateurs du cours IFT1169 ou bien durant la séance de cours du mardi 07 février 2006.

**Barème :** Ce TP1 est noté sur 6 points.

| | |
|---|---|
| **Compilation et respect des spécifications** | **0.5** |
| **Codage, commentaires etc.** | **1** |
| **Rapport** | **1.5** |
| **Tests (fournis)** | **3** |

**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.
- La non remise papier vous pénalise de 1 point.
- Les programmes ne contenant pas d'en-tête, -1 point.
- Un programme qui ne compile pas : 0.
- Un programme qui compile mais ne fait 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 :** C'est « l'ordinateur » qui va corriger en quelque sorte vos travaux. Ce dernier va utiliser des scripts qui vont prendre en considération les contraintes imposées. Pas d'espaces blancs en trop, pas de lignes en trop, pas de commentaires supplémentaires. Vous devez produire des sorties **identiques** aux fichiers fournis en exemples.

Pour s'assurer de cela, commencer d'abord par rediriger la sortie dans un fichier avec le symbole >. Comparer par la suite votre sortie avec la sortie de référence en utilisant pour cela la commande diff (sous MinGW/MSys sinon la commande fc sous DOS, ajuster juste les options) comme suit :

```
diff sortieref.txt votresortie.txt
```

Attention, les informations affichées sur la sortie des erreurs doivent être redirigées en utilisant la directive (2>&1) telle que décrite précédemment.

Ainsi donc au moment de la remise de votre travail, vous avez déjà une idée approximative de la note que vous allez obtenir. Si la commande « diff » ne signale pas de différences entre les deux fichiers, vous êtes sur la bonne voie pour obtenir une note élevée.

## Des questions à propos de ce TP?

Une seule adresse : pift1169@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 **tp01**.

## Mise à jour

30-01-2006 diffusion

## Annexe -1-

| Ligne de commande | Fichier de sortie |
|---|---|
| `tp1.exe` | `s01.txt` |
| `tp1.exe -a`<br>`tp1.exe -h` | `s02.txt` |
| `tp1.exe -v 2999` | `s03.txt` |
| `tp1.exe -a -h`<br>`tp1.exe -h -a` | `s04.txt` |
| `tp1.exe -v` | `s05.txt` |
| `tp1.exe -e` | `s06.txt` |
| `tp1.exe -a -v 999`<br>`tp1.exe -h -v 999`<br>`tp1.exe -a -h -v 999` | `s07.exe` |
| `tp1.exe toto`<br>`tp1.exe -a toto`<br>`tp1.exe -h toto`<br>`tp1.exe -v 999 toto` | `s08.txt` |

**Ce tableau décrit les seules lignes de commande qui seront prises en considération ainsi que les fichiers de sortie correspondants.**

## Annexe -2-

Les sorties « `s05.txt` » et « `s06.txt` » nécessitent de manipuler les variables « `opterr` » et « `optopt` » comme suit :

- Mettre la variable « `opterr` » à « `0` ». Ceci va permettre de désactiver l'affichage au format par défaut des erreurs. Nous allons donc éviter de recevoir par exemple le message en anglais « `option requires an argument – v` » signalant que nous avons omis d'inclure sur la ligne de commande la valeur associée à l'option « `v` ».

Puisque nous avons décidé de désactiver l'affichage automatique des erreurs, il faudra quand même faire le nécessaire pour les afficher manuellement! Pour ce faire :

- D'abord, il faudra se questionner : est-ce qu'il y a eu des erreurs de lecture? Plusieurs façons de le savoir : soit de vérifier le contenu du caractère retourné par « `getopt` », si c'est le caractère « `?` » il y a eu erreur ; ou bien d'attendre résolument les erreurs dans le bloc « `default` » associé au « `switch/case` ».

- En cas d'erreur, quel est le nom de l'option ayant généré cette erreur? Le nom de l'option soit inconnue soit dont la valeur a été omise, est préservée dans la variable « `optopt` ». Il suffit juste de la lire.

- Bien sûr, toute cette cuisine suppose que nous n'allons pas passer l'option « ? » sur la ligne de commande. Dans le cas contraire, il faudra appliquer une autre recette!