IFT1166 – TRAVAIL PRATIQUE #1 – 24 janvier

Le couteau suisse du C++!

Mohamed Lokbani

Équipes: Le travail est à faire en monôme (**une seule personne**).
Remise : Deux remises à effectuer : électronique et papier le **mercredi 02 février, 20h00 au plus tard, sans possibilité de retard**.
Conseil: N'attendez pas le dernier jour avant la remise pour commencer, vous n'aurez pas le temps.

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 C++. Nous allons introduire pour cela l'utilisation de la classe string, permettant de manipuler avec une certaine aisance les chaînes de caractères; ainsi que la manipulation des arguments passés en ligne de commande. Et finalement vous faire faire pratiquer la procédure de remise, la rédaction d'un rapport, le respect des échéances.

Énoncé : On se propose d'écrire un programme C++ permettant de lire de l'entrée standard une série de mots passés en ligne de commande et de les afficher par la suite sur la sortie standard dans un format préalablement fixé.

Caractéristiques techniques :

- **Format d'affichage :** correspond à l'une des justifications suivantes :
  - **g :** pour une justification à gauche.
  - **d :** pour une justification à droite.
  - **c :** pour une justification au centre.
- **Taille par défaut:** elle correspond nombre total de caractères à afficher. Pour ce travail, la taille par défaut est fixée à 20 caractères.
- **Remplissage :** elle définit le caractère de remplissage. Pour ce travail, ce caractère est **#**.

Vous pouvez ajouter à votre aise d'autres éléments si vous jugez cela nécessaire.

Scénarios entrée/sortie : Les éléments seront lus à partir de l'entrée standard. Nous allons d'abord préciser le format d'affichage (i.e. une des lettres g, d, c) puis la série de mots à lire. Ces paramètres sur la ligne de commande seront séparés par un espace blanc (voir le fichier fourni avec ce travail).

Après avoir lu ces informations, tout en tenant compte du format demandé, la série de mots sera affichée comme suit :

- si **g** : vous devez afficher la phrase à gauche et compléter à sa droite les champs vides par le caractère #.
- si **d** : vous devez afficher la phrase à droite et compléter à sa gauche les champs vides par le caractère #.
- si **c** : vous devez afficher la phrase au centre et compléter à sa droite et à sa gauche les champs vides par le caractère #. Si dans ce cas le nombre de champs vides est un nombre impair, le caractère # en trop sera placé à droite de la phrase.

Pour ce travail, la taille par défaut pour l'affichage en sortie a été fixée à seulement 20 caractères. Or il se peut que la série de mots ait une taille bien supérieure à la taille par défaut. Dans ce cas, il faut faire les ajustements nécessaires. La nouvelle taille (par défaut) doit être toujours un multiple de la taille par défaut. Par exemple si la série de mots à une taille totale de 24 (ou 38, 44, 78) caractères, espaces blancs entre les mots compris, la taille par défaut sera donc portée à 40 (resp. 40, 60, 80) caractères.

Hypothèses et contraintes :

- 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.
- IFT1166 est un cours sur le C++, donc votre programme doit être écrit en C++ et pas en C! 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>].
- Vous devez vérifier si les entrées sont correctes et afficher en conséquence les messages d'erreur appropriés (voir annexe -1-) :
    *) le nombre d'arguments est correct, i.e. avoir en entrée obligatoirement le format et la série de mots (devant contenir au moins un mot).
    *) un des formats doit être obligatoirement g, d, ou c et rien d'autre.
- Ce travail n'est pas un exercice d'algorithmique, donc pas besoin de compliquer un calcul simple!
- Assurer vous de respecter les noms des fichiers demandés, ainsi que le format de l'affichage en sortie.
- Des fichiers d'entrée/sortie sont fournis. On ne vous demande pas d'écrire du code pour lire/écrire à partir de fichiers, vous pouvez tester vos programmes à la main (en insérant manuellement les informations) comme vous pouvez utiliser les redirections (plus facile à manier et évite les erreurs de frappe). Les redirections sont au nombre de deux : < pour l'entrée et > pour la sortie. Exemple : [**tp1.exe < data.txt > masortie.txt**]. Dans cet exemple la lecture se fera à partir du fichier data.txt et l'écriture se fera dans le fichier masortie.txt.
- Les fichiers entrée/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.
- À signaler que nous utiliserons d'autres fichiers pour la correction. Si vous avez respecté les contraintes imposées, peu importe le jeu de test utilisé, il devra fournir normalement un résultat correct.

Remise : Il est important de noter que votre TP sera compilé avec gcc3.2. 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 du DESI. Avant de faire cela, assurez-vous d'abord que vous avez la bonne version de gcc, en utilisant pour cela la commande : "**gcc –v**" devra donnée le numéro de version "**3.2**".

La remise comprend **3** (**TROIS**) choses (avant le mercredi 02 février à 20h00) :

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 pift1166 travail01 tp1.cpp rapport

2. Remettez une **copie papier** d'un rapport qui devrait décrire votre programme (pour le contenu d'un rapport voir la FAQ sur la page web du cours).

3. N'oubliez pas de remettre avec votre rapport une **copie papier** de votre programme C++.

Barème : Ce TP1 est noté sur 4 points.

Compilation et respect des spécifications	**0.75**
Codage, commentaires etc.	**1.25**
Rapport	**0.75**
Tests (fournis et non fournis)	**1.25**

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 : 0/4.

- Un programme qui compile mais ne fait les choses prévues dans la spécification : 0/4.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.25 et plus.
- Le nom respect du nom de fichier, donc forcement il ne va pas compiler et un des points de la spécification n'a pas été respecté : 0/4.
- 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 tenir compte au détail près des contraintes imposées.
Rediriger la sortie dans un fichier avec le symbole > et la comparer au caractère près avec les sorties fournies en exemple, aider vous dans cette tâche de la commande diff (sous MinGW/MSys sinon la commande fc sous dos, ajuster juste les options) comme suit :

```
diff –b –w –i –B out1.txt votresortie.txt
```

Par ailleurs, nous utiliserons d'autres jeux de test pour la correction.

Des questions à propos de ce TP?

Une seule adresse : pift1166@iro.umontreal.ca

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre email, au moins la chaîne: [IFT1166] et une référence au **travail01**.

Mise à jour

24-01-2005 diffusion

Annexe -1-

Pour chaque test, nous avons fourni la ligne de commande en entrée (colonne 1) et le résultat obtenu (colonne 2). Nous avons fourni aussi des explications complémentaires pour chacun des tests. À signaler que chaque sortie est affichée obligatoirement entre **' '.**

Test	Entrée	Sortie	Commentaires
1	tp1.exe g ceci est exemple	'ceci est un exemple#'	g donc une justification à gauche donc le # est à droite de la phrase.
2	tp1.exe d ceci est exemple	'#ceci est un exemple'	d donc une justification à droite donc le # est à gauche de la phrase.
3	tp1.exe c ceci est exemple	'ceci est un exemple#'	c donc une justification au centre. La phrase compte 19 caractères, espaces compris. Taille par défaut étant 20, un seul # est permis. Donc nombre impair, il sera placé à droite de la phrase.
4	tp1.exe c ceci est un autre exemple plus grand	'##ceci est un autre exemple plus grand##'	c donc une justification au centre. La phrase compte 36 caractères, espaces compris. Taille par défaut a été ajustée à 40 (2x20), 4 # sont permis pour compléter la chaîne. Donc nombre pair, 2 # à droite et à gauche de la phrase.
5	tp1.exe ceci est un autre exemple plus grand	Erreur: justification inconnue!	Nous avons omis de préciser le format d'affichage.
6	tp1.exe c	Erreur: le nombre d'arguments doit être au moins égal à 3!	Nous avons omis de préciser la série de mots.

Annexe -2-

La ligne de commande qui lance l'exécution d'un programme peut contenir un ensemble d'arguments. Par exemple, la ligne de commande « tp1.exe d ceci est un exemple » est une ligne de commande qui contient 6 arguments.

Indice de l'argument	Argument	argv[]
0	tp1.exe	argv[0]
1	d	argv[1]
2	ceci	argv[2]
3	est	argv[3]
4	un	argv[4]
5	exemple	argv[5]

Dans les langages C et C++, l'argument 0 est le nom du programme (ce n'est pas le cas pour le langage java où le nom du programme n'est pas pris en compte). Il faut déclarer la fonction main de la manière suivante pour pouvoir accéder à ces arguments :

int main (int argc, char *argv[]) { //etc.}

argc : contient le nombre d'arguments de la ligne de commande.
argv est un tableau de pointeurs sur les arguments de la ligne de commande.

Ce simple programme affiche le nombre d'arguments et la liste des arguments de la ligne de commande :

```
#include <iostream>
using namespace std;

int main(int argc, char* argv[]){
    cout << "nbre d'arguments de la ligne de commande est: " << argc << endl;
    for (int i=0;i<argc;i++) cout << "argv[" << i << "]: " << argv[i] << endl;
    return 0;
}
```

Si la manipulation de pointeurs vous pause problème pour ce premier travail, vous pouvez convertir les chaînes de caractères, i.e char* vers le type string (voir annexe -3-).

Annexe -3-

En C++ la classe string permet de manipuler les chaînes de caractères sans trop se soucier de la gestion de pointeurs qui est faite par derrière. Le programme suivant présente quelques exemples d'utilisation du type string. À noter que pour pouvoir utiliser string il faut impérativement inclure le fichier d'entête « <string> ».

```
#include <iostream>
#include <string>

using namespace std;

int main() {

    string chaine, deuxchaines; // définition de 2 variables
    char tab[5]={'a','b','c','d','\0'}; // une chaîne de caractères sous la forme d'un tableau

    chaine = "ceci"; // initialisation d'une chaîne
    cout << "chaine au départ: " << chaine << endl;
    string uneautre = " est un exemple"; // définition et initialisation les 2 à la fois
    cout << "une autre chaine: " << uneautre << endl;
    chaine = chaine + uneautre; // on ajoute deux chaînes, une addition classique
    cout << "chaine après ajout: " << chaine << endl;
    cout << "la taille de ma chaine: " << chaine.length() << endl; // pour avoir la taille
    string untableau(tab); // définition et initialisation à l'aide d'un tableau
    cout << "untableau: " << untableau << endl;

    return 0;

}
```