

IFT1166 – TRAVAIL PRATIQUE #1 – 14 mai 2008**L'art de formater une sortie en C++!**

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 mai 2008, 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 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 dans ce travail l'utilisation des manipulateurs de format et ce, afin de vous faire pratiquer la procédure de remise, la rédaction d'un rapport, le respect des échéances.

Énoncé : on vous demande d'écrire dans un même programme C++ les instructions nécessaires permettant d'afficher en sortie des nombres dans divers formats. Vous allez utiliser pour cela les manipulateurs appropriés. Un manipulateur permet de préciser le format des données quand il est inséré dans les instructions de lecture (entrée) et d'écriture (sortie) d'un flot. Une description détaillée de ces manipulateurs se trouve en annexe -1-. Pour utiliser ces manipulateurs, il faudra inclure dans le programme le fichier <iomanip>.

Description des affichages : il est demandé de réaliser les affichages suivants (à faire dans l'ordre) :

Série -1-

```
nbre1 = 2, nbre2 = 2.0, nbre3 = 2.2, nbre3a = 2.22, nbre4 = 2.29, nbre5 = 2.222, nbre6 = 2.229,
nbre7 = 22, nbre8 = 222
```

Formats des affichages demandés en sortie

- affichage par défaut
- affichage par défaut et précision égale à 6
- affichage par défaut et précision égale à 2
- affichage par défaut, précision égale à 6 et virgule fixe

Série -2-

```
string unechaine = "ijklmn";
double undouble = 55.667;
```

Formats des affichages demandés en sortie

- affichage par défaut, et une largeur de 10 caractères pour les deux variables « unechaine » et « undouble »
- affichage par défaut, une largeur de 10 caractères, justification à gauche pour la variable « unechaine », et affichage par défaut, une largeur de 10 caractères, virgule fixe, justification à droite, et une précision égale à 1 pour la variable « undouble ».
- affichage par défaut, une largeur de 10 caractères, virgule fixe, justification à droite, et une précision égale à 1 pour la variable « undouble ».
- affichage par défaut, une largeur de 10 caractères, virgule fixe, justification à droite, et une précision égale à 4 pour la variable « undouble ».
- affichage par défaut, une largeur de 10 caractères, virgule fixe, justification à droite, et une précision égale à 8 pour la variable « undouble ».

- affichage par défaut, une largeur de 10 caractères, affichage scientifique, justification à droite, et une précision égale à 1 pour la variable « undouble ».
- affichage par défaut, une largeur de 10 caractères, affichage scientifique, justification à droite, et une précision égale à 4 pour la variable « undouble ».

Série -3-

double undouble = 66.;

Formats des affichages demandés en sortie

- affichage par défaut, une largeur de 10 caractères, et une précision égale à 2 pour la variable « undouble ».
- le point est affiché, et une largeur de 10 caractères pour la variable « undouble ».
- le point n'est pas affiché, et une largeur de 10 caractères pour la variable « undouble ».
- affichage par défaut, une largeur de 10 caractères, virgule fixe, et une précision égale à 4 pour la variable « undouble ».
- le point est affiché, et une largeur de 10 caractères pour la variable « undouble ».
- le point n'est pas affiché, et une largeur de 10 caractères pour la variable « undouble ».

double undouble = 66.1;

Formats des affichages demandés en sortie

- affichage par défaut, une largeur de 10 caractères, affichage scientifique, et une précision égale à 4 pour la variable « undouble ».
- le point est affiché, et une largeur de 10 caractères pour la variable « undouble ».
- le point n'est pas affiché, et une largeur de 10 caractères pour la variable « undouble ».

Série -4-

double undouble = 55.667;

double negundouble = -undouble;

Formats des affichages demandés en sortie

- virgule fixe, précision est égale à 2, et 10 caractères pour les deux variables « undouble » et « negundouble ».
- on affiche les positions, et 10 caractères pour les deux variables « undouble » et « negundouble ».
- on n'affiche plus les positions, et 10 caractères pour les deux variables « undouble » et « negundouble ».
- on fait du remplissage avec du « 0 », et 10 caractères pour les deux variables « undouble » et « negundouble ».
- justification droite, et 10 caractères pour les deux variables « undouble » et « negundouble ».
- justification gauche, et 10 caractères pour les deux variables « undouble » et « negundouble ».

Attention : lors du traitement d'une série vous devez vous assurer que toutes les spécifications ont été remises en l'état par défaut.

Fichier de sortie : le résultat obtenu suite à ces affichages se trouve dans le fichier « **sortieref.txt** ». Il est disponible sur la page web du cours. Il faudra vous aider de ce fichier pour organiser correctement les éléments à afficher.

Hypothèses et contraintes :

- pour commencer, vous devez déjà respecter les différentes contraintes de programmation précédemment décrites.
- IFT1166 est un cours en C++, donc votre programme doit être écrit en C++ et non 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 devez vous assurer de ne pas changer les noms des fichiers et de respecter par la même occasion le format de l'affichage en sortie.

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). Le rapport doit être au format « pdf » (voir là aussi la FAQ sur la page web du cours sur la manière de générer un fichier au format « pdf »).

Remise : il est important de noter que votre TP sera compilé avec gcc3.4.2. 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 "**3.4.2**".

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).

Si vous avez regroupé l'ensemble des fichiers dans un seul fichier « tp1.zip », vous devez faire la remise comme suit :

1. commencez d'abord par vous connecter sur la machine « remise » comme il a été pratiqué dans la démo #01.
2. envoyez l'ensemble de vos fichiers par la procédure de remise électronique habituelle (Pour obtenir de l'aide sur cette commande, tapez dans un Xterm : `man remise`). Respectez les noms des fichiers.

remise ift1166 tp1 tp1.zip

3. Vérifiez que la remise s'est effectuée correctement.

remise -v ift1166 tp1

Barème : Ce TP1 est noté sur 10 points.

Compilation et respect des spécifications	1
Codage, commentaires etc.	3
Rapport	1
Tests (fournis)	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 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!

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. Pas d'espaces blancs en trop, pas de commentaires supplémentaires. Vous devez produire une sortie **identique** à celle du fichier « **sortieref.txt** ».

Pour s'assurer de cela, commencez d'abord par rediriger la sortie dans un fichier avec le symbole >. Comparez 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 », ajustez juste les options. La commande « diff » permet donc de trouver les différences entre des fichiers. Si les fichiers sont identiques au caractère près, la commande ne retourne rien, sinon la liste des différences.

compilation	g++ -Wall -pedantic¹ -o tp1.exe tp1.cpp
redirection	tp1.exe > masortie.txt
comparaison	diff sortieref.txt masortie.txt

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 : 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 **tp01**.

Mise à jour

14-05-2008 diffusion

¹ L'éditeur « Scite » prend en compte par défaut de ces options.

Annexe -1-²**Manipulateurs de flux de sortie**

Manipulateur	Fonction
endl	Envoie un caractère de retour à la ligne sur le flux et synchronise le tampon par un appel à la méthode flush.
ends	Envoie un caractère nul terminal de fin de ligne sur le flux.
flush	Synchronise le tampon utilisé par le flux par un appel à la méthode flush.
boolalpha	Active le formatage des booléens sous forme textuelle.
noboolalpha	Désactive le formatage textuel des booléens.
hex	Formate les nombres en base 16.
oct	Formate les nombres en base 8.
dec	Formate les nombres en base 10.
fixed	Utilise la notation en virgule fixe pour les nombres à virgule.
scientific	Utilise la notation en virgule flottante pour les nombres à virgule.
left	Aligne les résultats à gauche.
right	Aligne les résultats à droite.
internal	Utilise le remplissage des champs avec des espaces complémentaires à une position fixe déterminée par la locale courante. Équivalent à right si la locale ne spécifie aucune position de remplissage particulière.
showbase	Indique la base de numérotation utilisée.
noshowbase	N'indique pas la base de numérotation utilisée.
showpoint	Utilise le séparateur de virgule dans les nombres à virgule, même si la partie fractionnaire est nulle.
noshowpoint	N'utilise le séparateur de virgule que si la partie fractionnaire des nombres à virgule flottante est significative.
showpos	Écrit systématiquement le signe des nombres, même s'ils sont positifs.
noshowpos	N'écrit le signe des nombres que s'ils sont négatifs.
uppercase	Écrit les exposants et les chiffres hexadécimaux en majuscule.
nouppercase	Écrit les exposants et les chiffres hexadécimaux en minuscule.
unitbuf	Effectue une opération de synchronisation du cache géré par le tampon du flux après chaque écriture.
nounitbuf	N'effectue les opérations de synchronisation du cache géré par le tampon du flux que lorsque cela est explicitement demandé.

Tableau -1- Manipulateurs sans paramètres

² Les tableaux 1 et 2 ont été extraits des notes de cours de Christian Casteyde ©

Manipulateur	Fonction
<code>resetiosflags (ios_base::fmtflags)</code>	Permet d'effacer certains bits des options du flux. Ces bits sont spécifiés par une combinaison logique de constantes de type <code>ios_base::fmtflags</code> .
<code>setiosflags (ios_base::fmtflags)</code>	Permet de positionner certains bits des options du flux. Ces bits sont spécifiés par une combinaison logique de constantes de type <code>ios_base::fmtflags</code> .
<code>setbase (int base)</code>	Permet de sélectionner la base de numérotation utilisée. Les valeurs admissibles sont 8, 10 et 16 respectivement pour la base octale, la base décimale et la base hexadécimale.
<code>setprecision (int)</code>	Permet de spécifier la précision (nombre de caractères significatifs) des nombres formatés.
<code>setw (int)</code>	Permet de spécifier la largeur minimale du champ dans lequel la donnée suivante sera écrite à la prochaine opération d'écriture sur le flux.
<code>setfill (char_type)</code>	Permet de spécifier le caractère de remplissage à utiliser lorsque la largeur du champ est inférieure à la largeur minimale spécifiée dans les options de formatage.

Tableau -2- Manipulateurs avec paramètres

Remarques importantes

-1- « `ios::fmtflags` » est une des constantes décrites dans le tableau -1-.

-2- Comme il est indiqué dans le tableau -2-, « `setiosflags` » peut combiner plusieurs constantes de la manière suivante : « `setiosflags (ios::noshowbase | ios::noboolalpha);` ». Ici on vient de désactiver les affichages de la base et le formatage textuel des booléens. On ne peut désactiver qu'un seul format (pas besoin de « `|` » dans ce cas) comme on peut désactiver plusieurs. Dans ce cas séparez chaque format par l'opérateur « `|` » qui correspond au « ou logique » mais appliqué au niveau des bits.

-3- Si un manipulateur n'est pas invoqué, le flot utilise la valeur par défaut ou bien la dernière valeur connue pour ce manipulateur. De ce fait, faites très attention! Si l'on ne veut pas utiliser les dernières valeurs connues des manipulateurs, il devient nécessaire de les remettre à leur état initial.