IFT1166 – TRAVAIL PRATIQUE #3 – 16 octobre 2005

Les caractères disséqués à la loupe par C++!

Mohamed Lokbani

Équipes: Le travail en équipe de deux est permis. Vous ne remettez alors qu'un travail par équipe.
Remise : Deux remises à effectuer : électronique et papier le mardi 08 novembre, 20h30 au plus tard, sans possibilité de retard. La copie papier doit-être remise à vos démonstrateurs ou pendant le cours du 08 novembre.
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 faire pratiquer les pointeurs. Pour cela, vous allez manipuler les chaînes de caractères en utilisant les rudiments de base du langage C++.

Énoncé : On se propose d'écrire un programme C++ permettant de saisir de l'entrée standard la paire: (taille, chaîne) où chaîne est une chaîne de caractères, et taille la longueur de cette chaîne. Ce programme doit proposer aussi à un utilisateur un menu permettant les opérations suivantes sur la chaîne saisie:

A- Sur la chaîne elle-même:

-1- Afficher la chaîne en question.
-2- Calculer la longueur de la chaîne.
-3- Inverser la chaîne.
-4- Calculer le nombre de mots[1] dans la chaîne.
-5- Supprimer les espaces blancs de la chaîne.
-6- Tester si la chaîne est un palindrome. Une chaîne est un palindrome si elle peut être lue indifféremment dans les deux sens en gardant la même signification. Par exemple le mot [ressasser] en est un, de même que les phrases [Elu par cette crapule].

B- À l'aide d'une seconde chaîne: Pour ces opérations, votre programme doit lire une seconde paire (taille, chaîne) en entrée.

-7- Ajouter à la première paire, la seconde. Les deux chaînes seront espacées par un espace blanc.
-8- Retirer tous les mots de la seconde paire de la première paire.
-9- Tester si les deux paires sont identiques. Par identique, nous entendons c'est exactement les mêmes chaînes de caractères.
-10- Tester si la chaîne est une anagramme d'une autre. Pour mémoire, une anagramme d'une chaîne est une chaîne formée des mêmes lettres que la première mais dans un ordre différent. Par exemple, les chaînes [Bison Ravi] ou [Bris avion] sont des anagrammes de [Boris Vian].

Contraintes: Vous devez utiliser les rudiments de base du langage C++.

- À ce stade du cours, nous n'avons pas vu **tous** les détails relatifs à l'utilisation des classes. Pour cette raison, il est préférable de vous contenter d'utiliser **que** les structures au lieu des classes. Donc il faut donner une attention particulière à la structure de votre programme. Vous êtes libres d'utiliser des classes … **mais à vos risques et périls**!

- Par contre, Il **ne** vous est pas permis d'utiliser le type **string, ni** des fonctions **STL** pour manipuler vos chaînes de caractères. Donc comme au bon vieux temps i.e. utiliser les « char* » et les fonctions associées (voir le fichier « `char911.cpp` »).

- Éviter (comme la peste) d'allouer vos tableaux de manière statique et privilégier plutôt l'allocation dynamique. Vous devez apprendre à être avare dans la gestion des ressources (mémoire etc.)!

[1] Un mot est une séquence des caractères qui ne contient aucun espace.

Scénario entrée/sortie : Il est ouvert pour ce travail pratique. Nous comptons ainsi sur votre imagination et votre créativité pour concevoir tous les détails de ce travail. Faire attention de ne pas oublier d'inclure dans votre rapport les différentes explications relatives à l'utilisation de votre programme ainsi que sa conception.

Jeux et Stratégies: Il est demandé de réaliser 10 scénarii. Vous pouvez les coder indépendamment, puis les connecter à la fin.

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 (qui normalement fonctionne correctement chez vous) fonctionne bien sur les ordinateurs du DESI. Pour avoir la version du compilateur utiliser la commande : "**gcc –v**" devra donnée le numéro de version "**3.4.2**".
Vous devez remettre 2 fichiers : « tp3.cpp » et « rapport ». Le fichier « tp3.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).
Pour ce travail, le rapport doit être au format « texte » ou « pdf ». Voir la rubrique « Foire Aux Questions » sur la page web du cours, sur la procédure à suivre pour produire des « pdf ».

La remise comprend **3** (**TROIS**) choses (**avant le mardi 08 novembre à 20h30**) :

1. Envoyez vos fichiers « tp3.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 ift1166 tp3 tp3.cpp rapport

2. Vérifier que la remise s'est effectuée correctement.
   remise –v ift1166 tp3

3. Remettez une **copie papier** de votre programme (« tp3.cpp ») ainsi que de votre rapport.

Barème : Ce TP3 est noté sur 13 points.

Compilation et respect des spécifications	**1**
Codage, commentaires etc.	**4**
Rapport	**3**
Tests	**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.
- 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 : 0/13.
- Un programme qui compile mais ne fait les choses prévues dans la spécification : 0/13.
- 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 : 0/13.
- Aberration dans le codage : Après 2 tps, il n'y a qu'un seul chemin qui mènera à Rome. Il passe par une écriture propre, claire, courte et bien structurée de votre programme. Éviter les fonctions (la fonction « main » comprise ») de « 10kms » de long ! Et svp pas d'usine à gaz (i.e. des choses compliquées pour rien !).
- Non respect des contraintes imposées ! Vous cherchez les ennuis ….

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]

Mise à jour

16-10-2005 Diffusion de l'énoncé.

Annexe -1-

Le fichier « **`char911.cpp`** » contient une description détaillée des fonctions utiles pour le traitement de caractères ou de chaînes de caractères.

Annexe 2

Encodage UTF-8

Qui doit lire cette annexe ?

Si :

1- Vous écrivez votre programme sur une machine Windows de la DESI en utilisant « Scite », « TextPad », « Notepad », « Emacs sous Windows » ou tout autre éditeur installé sous Windows.
2- Vous utilisez « Scite » ou « Mingw/Msys » ou « MSdotNet » pour compiler vos programmes.

Cette annexe ne vous concerne pas.

Par contre, si vous décidez d'éditer, (et/ou) compiler et (et/ou) exécuter vos travaux sur la machine « frontal » qui est une machine « Linux », vous devez lire attentivement ce qui suit.

C'est quoi « UTF-8 » ?

Les machines « Linux », du département utilise l'encodage Unicode « UTF-8 » par défaut. Une description détaillée de « UTF-8 » est disponible à cette adresse : http://fr.wikipedia.org/wiki/UTF-8

Cette adresse présente un manuel de survie dans un environnement « UTF-8 » sous Linux : http://www.tuteurs.ens.fr/faq/utf8.html

C'est quoi les incidences ?

L'utilisation de « UTF-8 » a une incidence sur l'affichage et la saisie des caractères sous « Linux », si vous décidez de travailler directement sur la machine « frontal ».

Pourquoi ?

Chaîne	**B**		**O**		**B**		**\0**	
ASCII (Hex sur 1 octet)	**42**		**6F**		**62**		**00**	
Unicode	**42**	**00**	**6F**	**00**	**62**	**00**	**00**	**00**
Représentation « Little Endian »	**00**	**42**	**00**	**6F**	**00**	**62**	**00**	**00**

La représentation des octets en machine est une source à problème. Imaginer deux machines ayant des processeurs différents, et ils décident d'arranger les octets différemment. Par exemple un processeur qui utilise la représentation « Little Endian » avec le format « Unicode », va représenter la chaîne « Bob » par « 0042 006F 0062 0000 ». Il y a donc inversion des octets à l'intérieur d'un même groupe. Au lieu d'avoir « 42 00 » pour « B », nous avons pour une telle représentation « 00 42 ».
Si nous devons utiliser la fonction « strlen », qui donne la longueur d'une chaîne, sur la version Unicode de la chaîne « Bob » pour une telle représentation, que va-t-il se passer ?
La fonction va retourner « 0 » comme longueur de la chaîne « Bob » ! Waouh !
Pourquoi ? Parce que le premier caractère rencontré est « 00 » qui sur un octet correspond au caractère fin de chaîne i.e. « \0 » ! La fonction « strlen » n'est pas appropriée pour manipuler cette chaîne écrite en « Unicode » sur un processeur utilisant « Little Endian » comme représentation.

La machine « frontal » utilise un tel processeur et l'encodage par défaut est « UTF-8 », donc des problèmes en perspective !

Que faire alors ?

Le but c'est de ne pas mixer les choses ; i.e. :

1- Travailler uniquement en « UTF-8 » : Éditer votre fichier en « UTF-8 » et n'utiliser que des fonctions équivalentes à celles décrites dans le fichier « **`char911.cpp`** », permettant de manipuler des caractères encodés en « UTF-8 ».

2- Travailler dans un autre encodage : Éditer votre fichier en « ISO-Latin1 » (format « classique ») et sélectionner dans votre programme[2] ou environnement de travail une structure de localisation du type « ISO-Latin1 ».

1. Travailler uniquement en « UTF-8 » :

Édition
La configuration de « Scite » actuelle permet d'éditer les fichiers dans le format « ISO-Latin1 ». Vous avez 3 options pour convertir ou forcer l'édition en mode « UTF-8 » :

- Modifier la configuration de « Scite » : Il faudra retirer le « # » de la ligne « #code.page=65001 » qui se trouve dans le fichier « SciTEGlobal.properties ». Ce fichier est localisé dans le répertoire d'installation de « Scite ».

- Recoder le fichier de « ISO-Latin1 » vers « UTF-8 ». Pour cela suivre les étapes décrites sur cette page : http://support.iro.umontreal.ca/recodage-en-utf8.shtml

- Utiliser un autre éditeur, « vi » par exemple, en faisant attention d'avoir un terminal préconfiguré pour supporter « UTF-8 ». Par défaut les terminaux ouverts à partir d'une station Linux sont préconfigurés pour supporter « UTF-8 ». Par contre, si vous utilisez l'émulateur de terminal « Putty[3] », vous devez vous assurer qu'il est correctement configuré. Voir sur cette page la procédure à suivre : http://support.iro.umontreal.ca/utf8-win.shtml

Fonctions
Le tableau suivant présente les fonctions décrites dans le fichier « **`char911.cpp`** » et leur équivalence en « Unicode » :

« Classique »	Unicode
char	wchar_t
strXXXX (srcpy, strlen …)	wcsXXXX (wcscpy, wcslen …)
isxxx (isalpha …)	iswxxxx (iswalpha ….)
ctype	wctype
string.h	cwchar

wchar_t Unechaine = L"Ceci est un test" ;
Il faudra ajouter la lettre « L » pour signifier que la chaîne constante qui suit la lettre est codée en Unicode.

2. Travailler dans un autre encodage :
Vous aimeriez travailler sur frontal dans un encodage autre que « UTF-8 », par exemple l'encodage « classique » i.e. « ISO-Latin1 » ; suivre pour cela les étapes décrites sur cette page : http://support.iro.umontreal.ca/isolatin1.shtml
Assurez-vous que votre éditeur est configuré pour ouvrir un document dans un encodage « ISO-Latin1 ».

[2] Nous n'allons pas décrire cette étape car elle dépasse le cadre de ce cours.
[3] En réalité il est appelé par défaut si vous cliquez sur « Raccourcis DESI \ Branchement sur UNIX ».