IFT1166 – TRAVAIL PRATIQUE #2 – 02 février 2004

C++ sur les traces du mot mystère!

Mohamed Lokbani

Équipes: Le travail est à faire en monôme ou en binôme, mais pas plus de deux. Vous ne remettez qu'un seul travail par équipe.
Remise : Deux remises à effectuer : électronique et papier le **mercredi 23 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 faire pratiquer l'utilisation des structures, des fonctions, de la compilation séparée et finalement la séparation en fichier .h (entête) et .cpp (source), en utilisant pour cela les notions de C++ étudiées jusqu'à présent.

Énoncé : Le jeu du mot caché est constitué d'une grille et d'une liste de mots. Le but du jeu est de repérer dans la grille chacun des mots de la liste. Si le mot a été localisé, il faudra encercler ses lettres dans la grille, puis le rayer de la liste des mots. Quand la liste de mots sera épuisée, seules les lettres non encerclées formeront le mot caché. Vous pouvez vous aider d'un livre de jeux, disponible chez un marchand de journaux, pour vous pratiquer à ce jeu.

Caractéristiques techniques : Ayant une grille et une liste de mots, vous avez à écrire un programme en C++ permettant de trouver le mot caché. Le programme doit respecter les spécifications décrites dans ce travail pratique. Le format de l'affichage en sortie doit correspondre exactement au format demandé.

- Caractéristiques de la grille: La grille est carrée. Les données de cette grille sont stockées dans un fichier. La première ligne de ce fichier donne la dimension de la grille. Les caractères de la grille peuvent être en majuscule ou en minuscule et ils ne sont pas accentués.

- Caractéristiques des mots de la liste: Les mots de la liste seront placés dans la grille dans les deux sens horizontaux, dans les deux sens verticaux et dans les quatre sens diagonaux. Un mot de la liste peut se retrouver zéros ou plusieurs fois dans la grille. Les caractères de ce mot, peuvent être en majuscule ou en minuscule et ils ne sont pas accentués. Cette liste de mots est stockée dans un fichier. La première ligne de ce fichier donne le nombre de mots dans la liste.

Fichiers fournis : Pour réaliser ce travail, nous fournissons les fichiers suivants, disponibles à partir de la page web du tp2 :

Nom de fichier	Description
tp2H05.cpp	Le programme principal
data.cpp et data.h	Un ensemble de fonctions nécessaires pour la saisie des données
motcache.cpp et motcache.h	Doivent contenir les fonctions à réaliser
Une série d'exemples	Des exemples pour tester votre travail

Description du fichier de sortie : La grille est représentée dans un espace (X, Y). Comme nous l'avons déjà mentionné, un mot peut se retrouver dans 8 directions possibles. Ces direction sont représentées dans l'ordre suivant: {nord, nordest, est, sudest, sud, sudouest, ouest, nordouest}. C'est cet ordre (le sens des aiguilles d'une montre) qu'il faudra utiliser lors de la procédure de recherche d'un mot de la liste dans la grille.

Suite à votre recherche, les cas suivants peuvent se présenter:

- un mot de la liste se retrouve une seule fois dans la grille: Si par exemple "amateur" est le mot recherché, dans ce cas l'affichage en sortie devra être comme suit:

Trouve le mot: amateur | position: (12,6) | Direction: est

La première lettre du mot est localisée à la 12e ligne et la 6e colonne. Le mot se retrouve vers la direction est à partir de cette position (12,6).

- un mot de la liste se retrouve plusieurs fois dans la grille: Si par exemple "amateur" est le mot recherche, et si ce mot se retrouve 3 fois dans la grille, dans ce cas l'affichage en sortie devra être comme suit:

Trouve le mot: amateur | position: (12,6) | Direction: nord
Trouve le mot: amateur | position: (12,6) | Direction: est
Trouve le mot: amateur | position: (1,1) | Direction: sud
le mot a ete trouve 3 fois.

- **un mot de la liste n'a pas été retrouvé dans la grille**: Si par exemple "amateur" est le mot recherché et il n'y a aucune trace de ce mot dans la grille, dans ce cas l'affichage en sortie devra être comme suit:

amateur: non trouve!

Même si le mot n'a pas été trouvé dans la grille, le programme poursuit quand même son travail à la recherche du mot suivant dans la liste des mots.

- format d'affichage des mots: les mots doivent être affichés dans leur format d'origine i.e. celui du fichier mots.txt.

- format d'affichage du mot caché: Si par exemple "jeux" est le mot caché, dans ce cas l'affichage en sortie devra être comme suit:

Le mot mystere est: JEUX

Contraintes algorithmiques : Nous n'imposons aucune approche pour résoudre ce problème. Vous devez utiliser nos fonctions pour lire les données, par la suite vous devez organiser ces données dans une (ou plusieurs) structure(s) adéquate(s) à ce problème.

Contraintes Techniques :

- 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 : [**tp2.exe gille.txt mots.txt > masortie.txt**]. Dans cet exemple 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.

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.

Si les fichiers sont identiques, vous n'obtenez rien en sortie. Sinon, vous obtiendrez les différences entre les deux fichiers. Pour plus de détails sur cette commande unix, sur votre console xterm, exécutez la commande suivante: **man diff**.

Compilation : Pour générer l'exécutable, tp2.exe, vous pouvez utiliser par exemple une des solutions suivantes :

-1- Utiliser le fichier Makefile fourni avec ce travail. Lire le contenu du fichier pour comprendre comment l'utiliser.

-2- Faire une compilation séparée à la main :

a) compiler le fichier motcache.cpp à part (ou CTRL-F7 sous scite) :

```
g++ -c motcache.cpp -Wall -pedantic -Os -I/. -L./
```

b) compiler le fichier data.cpp à part (ou CTRL-F7 sous scite) :

```
g++ -c data.cpp -Wall -pedantic -Os -I/. -L./
```

c) compiler le fichier tp2H05.cpp à part (ou CTRL-F7 sous scite) :

```
g++ -c tp2H05.cpp -Wall -pedantic -Os -I/. -L./
```

d) générer l'exécutable (ou F7 sous scite):

```
g++ -o tp2.exe tp2H05.o data.o motcache.o -Wall -pedantic -Os -I/. -L./
```

tp2.exe sera le programme exécutable. Ne pas oublier de mettre dans le même répertoire les fichiers *.h et *.cpp pour que la compilation puisse avoir lieu correctement.
En utilisant scite, la touche F7 permet de construire le projet. Elle utilise pour cela le fichier Makefile s'il est présent dans le répertoire. Donc faire attention à ce détail ! Placer le fichier Makefile dans un autre répertoire si vous ne voulez pas l'utiliser.

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 23 février à 20h00) :

1. Envoyez vos fichiers « motcache.cpp » « motcache.h » 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 travail02 motcache.cpp motcache.h 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 12 points.

Compilation et respect des spécifications	**1**
Algorithmique, codage, commentaires etc.	**4**
Rapport	**3**
Tests (fournis et non fournis)	**4**

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 : 0/12.
- Un programme qui compile mais ne fait les choses prévues dans la spécification : 0/12.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.5 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 : 0/12.
- Aberration dans le codage : Même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long! -0.5 et plus, en fonction des dégâts constatés !

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 **travail02**.

Mise à jour

02-02-2005 diffusion

Questions et Réponses

Questions Pratiques

-Q : En quoi consiste notre travail au juste?
-R : Ayant en sa possession une grille et une liste de mots, il faudra trouver le mot mystère. Vous devez donc écrire le code nécessaire pour trouver ce mot mystère (caché).

-Q : Où doit-on écrire nos programmes?
-R : Les fichiers motcache.h et motcache.cpp

-Q : Est-ce que je dois écrire une fonction main ?
-R : Non, puisqu'on fournit une. Voir le fichier tp2H05.cpp.

-Q : Est-ce que je peux modifier alors le contenu du fichier tp2H05.cpp ?
-R : Non vous ne pouvez pas modifier le fichier tp2H05.cpp. Ceci écrit, vous pouvez quand même écrire un pseudo fichier montp2.cpp qui va contenir votre fonction main test et ceci dans le processus de développement uniquement. Avant de remettre votre travail assurez-vous de compiler avec la version originale fournie, i.e. tp2H05.cpp. C'est cette version et rien d'autre que nous allons utiliser lors de la correction.

Q : Qu'est-ce qu'il en est pour les fichiers data.h et data.cpp. Est-ce que je peux les modifier ?
R : Non, il ne vous est permis de les modifier. Ces fichiers constituent l'interface avec les fichiers de données (par exemple grille.txt et mots.txt). Nous allons tester aussi vos programmes avec des données autres que celles fournies. Il nous faut nos fonctions d'entrée/sortie décrites dans data.h et data.cpp pour lire ces données.

-Q : Je ne comprends pas le code développé dans les fichiers data.h et data.cpp ? Est-ce normal ?
-R : Oui c'est tout à fait normal. Nous allons étudier la manipulation des fichiers plus tard dans le cours. Pas besoin de vous attarder sur le comment ont été codées les fonctions de lecture. Il faudra vous concentrer plutôt sur la manière de les utiliser.

-Q : L'énoncé n'est pas clair et j'ai besoin d'aide ! Comment procéder ?
-R : Nous sommes disponibles suivant l'horaire affiché sur la page web du cours, rubrique « Professeurs et démonstrateurs ». Vous pouvez aussi nous envoyer vos questions par courriel.

-Q : Mais je ne pourrai pas être présent aux horaires indiqués ?
-R : Envoyer nous un courriel pour voir s'il nous est possible de vous rencontrer à un autre moment.

Q : Pouvez-vous m'expliquer ce qui ne va pas? Je vous envoie pour cela mes fichiers motcache.cpp et motcache.h !
R : N'envoyer rien par courriel. Privilégier plutôt les séances de disponibilité offertes à vous.

-Q : Où se trouvent les fichiers grille.txt, et mots.txt ?
-R : Sur la page web du cours, rubrique « Démonstrations et devoirs », puis « tp2 ».

Questions Techniques générales

-Q : C'est quoi au juste unsigned?
-R : Si je prends l'exemple du type short et soit la déclaration suivante :

short x;
Un short par exemple est codé sur 16 bits. Le bit le plus fort (le 16e) est le bit de signe et les 15 autres bits servent à coder la valeur. Ainsi une variable du type short peut prendre une valeur comprise entre: -32768 et 32767.

unsigned short x;
Cette déclaration signifie que la variable n'est plus signée, i.e. le 16e bit ne servira plus comme bit de signe, ainsi la représentation de la variable est étendue et passe donc de 0 à 65535.

-Q : C'est quoi int main(int argc, char** argv) ?
-R : Voir le tp#1

-Q : Pourquoi avoir divisé le code entre deux fichiers *.h et *.cpp ?
-R : .h est le fichier d'en-tête. Il ne contient que les déclarations des fonctions. Ce fichier permet la factorisation des déclarations. Le fichier .cpp est le fichier d'implémentation. Il contient le codage (la définition) des fonctions.

-Q : Doit-on faire le tp2 avec des classes et objets?? Où devons nous nous contenter d'utiliser des structures et des fonctions ?
-R : Vous ne devez pas utiliser de classes & objets pour faire ce travail. La raison est toute simple, vous n'êtes pas assez avancés en cours. Vous aurez l'occasion de coder des classes dans les prochains travaux. Ceci écrit, vous devez par contre faire une conception orientée objet, de ce fait les structures (et fonctions) est l'approche la mieux appropriée.

Questions Algorithmiques et autres

-Q : Un mot peut-il être sur un seul caractère et si c'est le cas, comment indiquer la direction?
-R : Non, un mot ne peut pas être sur un seul caractère.

-Q : Peut-on utiliser qu'un seul tableau pour faire le tp?
-R : Chacun a sa manière de coder. Ça sera donc au cas par cas. Voir plutôt les critères de correction sur l'énoncé du tp.

-Q : À propos de « char* table[MAX]; » dans « load_data(table, taille,argv1); ». J'ai suivi les instructions et j'ai tenté de trouver dans les notes de cours où il est question d'un tableau de pointeurs, Je ne trouve rien ! Pourriez-vous m'indiquer des liens utiles ?
-R : Vous trouverez des détails sur les pointeurs dans un livre de C par exemple. Vous pouvez aussi jeter un coup d'œil à l'url suivant, chapitre 10, « Pointeurs et tableaux » :
http://www-inf.int-evry.fr/COURS/COURSC/index.html

Q : Le mot mystère est: JEUX », doit-on nécessairement afficher le mot mystère en majuscule, ou tout simplement dans la même casse que les caractères du fichier grille.txt ?
R : Oui, vous devez l'afficher en majuscule.

Q : On pourrait obtenir des résultats légèrement différents dépendant de l'ordre des directions dans lequel on vérifie la présence des mots. Par exemple, en vérifiant d'abord dans la direction NORD, suivi du SUD etc. versus NORD, suivi de l'EST, etc. Devrait-on utiliser le même ordre que celui de la liste fournie dans l'énoncé?
R : Oui, vous devez utiliser le même ordre. Pour une position donnée, vous devez donc rechercher dans le sens des aiguilles du montre i.e. {nord, nordest, est, sudest, sud, sudouest, ouest, nordouest}.

Q : Des lettres déjà "utilisées" peuvent-elles servir à nouveau? Par exemple si le mot "amateur" a été trouvé dans la liste, ces lettres peuvent-elles encore servir pour trouver des mots comme "mat", "urbain", etc.
R : Oui, les lettres utilisées peuvent servir à nouveau. Pour avoir le cœur net, faire à la main la première grille.

Q : La compilation se passe bien, mais une faute de segmentation apparaît à l'exécution?
R : "table" a été déclarée comme un tableau de pointeurs. Chaque pointeur devra pointer UNE CHAÎNE de caractères. Si vous ne prévoyez pas de l'espace mémoire pour contenir cette chaîne, vous êtes entrain d'utiliser un espace non alloué, et le programme finira par se planter à l'exécution car vous utilisez quelque chose qui n'existe pas. C'est la raison pour laquelle vous recevez le message « segmentation fault ».