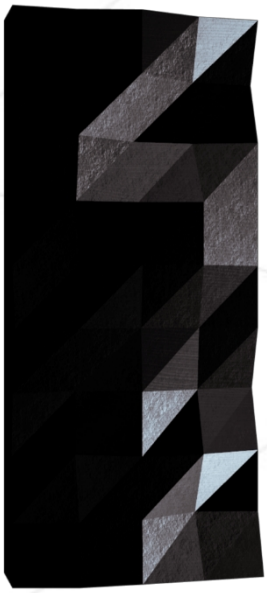

C++ - Module 09

STL

Résumé :
Ce document contient les exercices du module 09 des modules C++.

Version : 2

Contenu

Chapitre I

Introduction

Le C++ est un langage de programmation généraliste créé par Bjarne Stroustrup en tant qu'ex tension du langage de programmation C, ou "C avec des classes" (source : Wikipedia).

L'objectif de ces modules est de vous initier à la **programmation orientée objet**. Ce sera le point de départ de votre voyage en C++. De nombreux langages sont recommandés pour apprendre la POO. Nous avons décidé de choisir le C++ car il est dérivé de votre vieil ami le C. Comme il s'agit d'un langage complexe, et afin de garder les choses simples, votre code sera conforme à la norme C++98.

Nous sommes conscients que le C++ moderne est très différent à bien des égards. Si vous voulez devenir un développeur C++ compétent, il ne tient qu'à vous d'aller plus loin après le 42e tronc commun !

Chapitre II Règles générales

Compilation

- Compilez votre code avec c++ et les options -Wall -Wextra -Werror
- Votre code devrait toujours être compilé si vous ajoutez le drapeau -std=c++98

Conventions de formatage et de dénomination

- Les répertoires d'exercices seront nommés de la manière suivante : ex00, ex01, ... , exn
- Nommez vos fichiers, classes, fonctions, fonctions membres et attributs comme l'exigent les lignes directrices.
- Les noms des classes sont écrits en **majuscules**. Les fichiers contenant du code de classe seront toujours nommés en fonction du nom de la classe. Par exemple : Nomdeclasse.hpp/Nomdeclasse.h, Nomdeclasse.cpp, ou Nomdeclasse.tpp. Ainsi, si vous avez un fichier d'en-tête contenant la définition d'une classe "BrickWall" représentant un mur de briques, son nom sera BrickWall.hpp.
- Sauf indication contraire, tous les messages de sortie doivent être terminés par un caractère de nouvelle ligne et affichés sur la sortie standard.
- *Au revoir Norminette !* Aucun style de codage n'est imposé dans les modules C++. Vous pouvez suivre votre style préféré. Mais gardez à l'esprit qu'un code que vos pairs évaluateurs ne peuvent pas comprendre est un code qu'ils ne peuvent pas noter. Faites de votre mieux pour écrire un code propre et lisible.

Autorisé/interdit

Vous ne codez plus en C. Il est temps de passer au C++ ! C'est pourquoi :

- Vous êtes autorisé à utiliser presque tout ce qui se trouve dans la bibliothèque standard. Ainsi, au lieu de vous en tenir à ce que vous connaissez déjà, il serait judicieux d'utiliser autant que possible les versions C++ des fonctions C auxquelles vous êtes habitué.
- Cependant, vous ne pouvez pas utiliser d'autres bibliothèques externes. Cela signifie que les bibliothèques C++11 (et formes dérivées) et Boost sont interdites. Les fonctions suivantes sont également interdites : *printf(), *alloc() et free(). Si vous les utilisez, votre note sera de 0 et c'est tout.

- Notez que, sauf indication contraire, les espaces de noms d'utilisation <ns_name> et les mots-clés amis sont interdits. Dans le cas contraire, votre note sera de -42.
- **Vous n'êtes autorisé à utiliser le STL que dans les modules 08 et 09.** Cela signifie : pas de **conteneurs** (vector/list/map/etc.) et pas d'**algorithmes** (tout ce qui nécessite d'inclure l'en-tête <algorithm>) jusqu'à cette date. Sinon, votre note sera de -42.

Quelques exigences en matière de conception

- Les fuites de mémoire se produisent également en C++. Lorsque vous allouez de la mémoire (en utilisant la fonction new
), vous devez éviter les **fuites de mémoire.**
- Du module 02 au module 09, vos cours doivent être conçus selon la **forme canonique orthodoxe, sauf indication contraire explicite.**
- Toute implémentation de fonction placée dans un fichier d'en-tête (à l'exception des modèles de fonction) signifie 0 pour l'exercice.
- Vous devez pouvoir utiliser chacun de vos en-têtes indépendamment des autres. Ils doivent donc inclure toutes les dépendances dont ils ont besoin. Cependant, vous devez éviter le problème de la double inclusion en ajoutant des **gardes d'inclusion**. Dans le cas contraire, votre note sera de 0.

Lisez-moi

- Vous pouvez ajouter des fichiers supplémentaires si nécessaire (par exemple, pour diviser votre code). Comme ces travaux ne sont pas vérifiés par un programme, vous êtes libre de le faire tant que vous rendez les fichiers obligatoires.
- Parfois, les lignes directrices d'un exercice semblent courtes, mais les exemples peuvent montrer des exigences qui ne sont pas explicitement écrites dans les instructions.
- Lisez entièrement chaque module avant de commencer ! Vraiment, faites-le.
- Par Odin, par Thor ! Utilisez votre cerveau ! !!

Vous devrez implémenter un grand nombre de classes. Cela peut sembler fastidieux, à moins que vous ne soyez capable d'utiliser votre éditeur de texte favori.

Vous disposez d'une certaine liberté pour réaliser les exercices. Toutefois, respectez les règles obligatoires et ne soyez pas paresseux. Vous passeriez à côté d'un grand nombre d'informations utiles ! N'hésitez pas à vous documenter sur les concepts théoriques.

Chapitre III

Règles spécifiques aux modules

Il est obligatoire d'utiliser les conteneurs standard pour réaliser chaque exercice de ce module.

Une fois qu'un conteneur est utilisé, vous ne pouvez plus l'utiliser pour le reste du module.

Il est conseillé de lire le sujet dans son intégralité avant de faire les exercices.

Vous devez utiliser au moins un récipient pour chaque exercice, à l'exception de l'exercice 02 qui nécessite l'utilisation de deux récipients.

Vous devez soumettre un Makefile pour chaque programme qui compilera vos fichiers sources vers la sortie requise avec les drapeaux -Wall, -Wextra et -Werror.

Vous devez utiliser c++, et votre Makefile ne doit pas relinker.

Votre Makefile doit au moins contenir les règles $(NAME), all, clean, fclean et re.

Chapitre IV

Exercice 00 : Bourse de Bitcoin

	Exercice : 00
Bourse de Bitcoin	
Annuaire de retournement : *ex00/*	
Fichiers à rendre : `Makefile, main.cpp, BitcoinExchange.{cpp, hpp}`	
Fonctions interdites : `Aucune`	

Vous devez créer un programme qui calcule la valeur d'un certain montant de bitcoins à une certaine date.

Ce programme doit utiliser une base de données au format csv qui représentera le prix du bitcoin dans le temps. Cette base de données est fournie avec ce sujet.

Le programme prend en entrée une seconde base de données, qui stocke les différents prix/dates à évaluer.

Votre programme doit respecter ces règles :

- Le nom du programme est btc.
- Votre programme doit prendre un fichier en argument.
- Chaque ligne de ce fichier doit respecter le format suivant : "date | valeur".
- Une date valide se présente toujours sous le format suivant : Année-Mois-Jour.
- Une valeur valide doit être soit un flottant, soit un entier positif, entre 0 et 1000.

Vous devez utiliser au moins un conteneur dans votre code pour valider cet exercice. Vous devez gérer les erreurs éventuelles avec un message d'erreur approprié.

Voici un exemple de fichier input.txt :

```
$> head input.txt
date | valeur
2011-01-03 | 3
2011-01-03 | 2
2011-01-03 | 1
2011-01-03 | 1.2
2011-01-09 | 1
2012-01-11 | -1
2001-42-42
2012-01-11 | 1
2012-01-11 | 2147483648
$>
```

Votre programme utilisera la valeur de votre fichier d'entrée.

Votre programme doit afficher sur la sortie standard le résultat de la valeur multipliée par le taux de change en fonction de la date indiquée dans votre base de données.

Si la date utilisée dans l'entrée n'existe pas dans votre base de données, vous devez utiliser la date la plus proche contenue dans votre base de données. Veillez à utiliser la date inférieure et non la date supérieure.

Voici un exemple d'utilisation du programme.

```
$> ./btc
Erreur : Impossible d'ouvrir le fichier.
$> ./btc input.txt
2011-01-03 => 3 = 0.9
2011-01-03 => 2 = 0.6
2011-01-03 => 1 = 0.3
2011-01-03 => 1.2 = 0.36
2011-01-09 => 1 = 0.32
Erreur : pas un nombre positif.
Erreur : mauvaise entrée => 2001-
42-42 2012-01-11 => 1 = 7.1
Erreur : nombre trop grand.
$>
```

Attention : Le(s) conteneur(s) utilisé(s) pour valider cet exercice ne sera(ont) plus utilisable(s) pour la suite de ce module.

Chapitre V

Exercice 01 : Notation polonaise inversée

	Exercice : 01
RPN	
Répertoire de retournement : *ex01/*	
Fichiers à rendre : `Makefile, main.cpp, RPN.{cpp, hpp}`	
Fonctions interdites : `Aucune`	

Vous devez créer un programme avec ces contraintes :

- Le nom du programme est RPN.
- Votre programme doit prendre une expression mathématique polonaise inversée comme argument.
- Les nombres utilisés dans cette opération et passés en argument seront toujours inférieurs à 10. Le calcul lui-même ainsi que le résultat ne tiennent pas compte de cette règle.
- Votre programme doit traiter cette expression et afficher le résultat correct sur la sortie standard.
- Si une erreur survient pendant l'exécution du programme, un message d'erreur doit être affiché sur la sortie standard.
- Votre programme doit être en mesure de traiter des opérations avec ces jetons : "+ - / *".

`Vous devez utiliser au moins un conteneur dans votre code pour valider cet exercice.`

Vous n'avez pas besoin de gérer les parenthèses ou les nombres décimaux.

Voici un exemple d'utilisation standard :

```
$> ./RPN "8 9 * 9 - 9 - 9 - 4 - 1 +"
42
$> ./RPN "7 7 * 7 -"
42
$> ./RPN "1 2 * 2 / 2 * 2 4 - +"
0
$> ./RPN "(1 + 1)"
Erreur
$>
```

Attention : Le(s) conteneur(s) que vous avez utilisé(s) dans l'exercice précédent est (sont) interdit(s) ici. Le(s) conteneur(s) que vous avez utilisé(s) pour valider cet exercice ne sera(ont) pas utilisable(s) pour la suite de ce module.

Chapitre VI

Exercice 02 : PmergeMe

	Exercice : 02
PmergeMe	
Annuaire de retournement : *ex02/*	
Fichiers à rendre : `Makefile, main.cpp, PmergeMe.{cpp, hpp}`	
Fonctions interdites : `Aucune`	

Vous devez créer un programme avec ces contraintes :

- Le nom du programme est PmergeMe.
- Votre programme doit pouvoir utiliser une séquence d'entiers positifs comme argument.
- Votre programme doit utiliser l'algorithme de tri par fusion-insertion pour trier la séquence d'entiers positifs.

`Pour clarifier, oui, vous devez utiliser l'algorithme de Ford-Johnson.`

- Si une erreur se produit pendant l'exécution du programme, un message d'erreur doit être affiché sur la sortie standard.

`Vous devez utiliser au moins deux conteneurs différents dans votre code pour valider cet exercice. Votre programme doit pouvoir gérer au moins 3000 entiers différents.`

Il est fortement conseillé d'implémenter votre algorithme pour chaque conteneur et donc d'éviter d'utiliser une fonction générique.

Voici quelques indications supplémentaires sur les informations à afficher ligne par ligne sur la sortie standard :

- Sur la première ligne, vous devez afficher un texte explicite suivi de la séquence d'entiers positifs non triés.
- Sur la deuxième ligne, vous devez afficher un texte explicite suivi de la séquence d'entiers positifs triés.
- Sur la troisième ligne, vous devez afficher un texte explicite indiquant le temps utilisé par votre algorithme en spécifiant le premier conteneur utilisé pour trier la séquence d'entiers positifs.
- Sur la dernière ligne, vous devez afficher un texte explicite indiquant le temps utilisé par votre algorithme en spécifiant le deuxième conteneur utilisé pour trier la séquence d'entiers positifs.

Le format d'affichage du temps utilisé pour effectuer votre tri est libre mais la précision choisie doit permettre de voir clairement la différence entre les deux conteneurs utilisés.

Voici un **exemple** d'utilisation standard :

```
$> ./PmergeMe 3 5 9 7 4
Avant : 3 5 9 7 4
Après :  3 4 5 7 9
Temps de traitement d'une plage de 5 éléments avec std: :[..] :
0.00031 us Temps de traitement d'une plage de 5 éléments avec std:
:[...] : 0.00014 us
$> ./PmergeMe `shuf -i 1-100000 -n 3000 | tr "\n" " "`
Avant : 141 79 526 321 [...]
Après :  79 141 321 526 [...]
Temps de traitement d'une plage de 3000 éléments avec std: :[...] : 62.14389 us
Temps de traitement d'une plage de 3000 éléments avec std: :[...] : 69.27212
us
./PmergeMe "-1" "2" Erreur
$> # Pour OSX USER :
$> ./PmergeMe `jot -r 3000 1 100000 | tr '\n' ' '`
[...]
$>
```

L'indication de l'heure est volontairement étrange dans cet exemple. Il est évident que vous devez indiquer le temps utilisé pour effectuer toutes vos opérations, tant pour la partie tri que pour la partie gestion des données.

Attention : Le(s) récipient(s) que vous avez utilisé(s) dans les exercices précédents est (sont) interdit(s) ici.

La gestion des erreurs liées aux doublons est laissée à votre discrétion.

Chapitre VII

Soumission et évaluation par les pairs

Rendez votre travail dans votre dépôt Git comme d'habitude. Seul le travail contenu dans votre dépôt sera évalué lors de la soutenance. N'hésitez pas à vérifier les noms de vos dossiers et fichiers pour vous assurer qu'ils sont corrects.

16D85ACC441674FBA2DF65190663F33F793984B142405F56715D5225FBAB6E3D6A4F
167020A16827E1B16612137E59ECD492E47AB764CB10B45D979615AC9FC74D521D9
20A778A5E