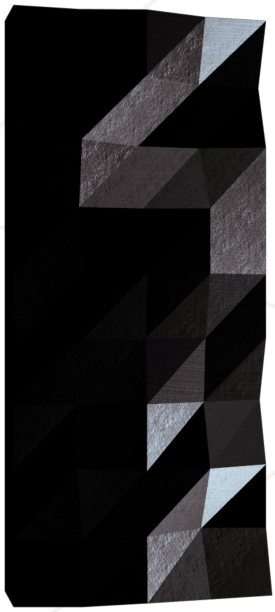

C++-Module 09

LIST

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

Version 1

Contenu

Chapitre I

Introduction

C++ est un langage de programmation à usage général créé par Bjarne Stroustrup comme une extension du langage de programmation C, ou "C with Classes" (source :Wikipédia).

L'objectif de ces modules est de vous présenter**Programmation orientée objet**. Ce sera le point de départ de votre parcours C++. De nombreuses langues sont recommandées pour apprendre la POO. Nous avons décidé de choisir C++ car il est dérivé de votre vieil ami 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. Alors si vous voulez devenir un développeur C++ compétent, à vous d'aller plus loin après le 42 Common Core !

Chapitre II

Règles générales

Compilation

- Compilez votre code avecc++et les drapeaux -Mur -Wextra -Werror
- Votre code devrait toujours être compilé si vous ajoutez le drapeau -std=c++98

Conventions de formatage et de nommage

- Les répertoires d'exercices seront nommés ainsi :ex00, ex01, ... , exn
- Nommez vos fichiers, classes, fonctions, fonctions membres et attributs comme requis dans les instructions.
- Écrivez les noms de classe dans**MajusculeCamelCase**format. Les fichiers contenant le code de la classe seront toujours nommés en fonction du nom de la classe. Par exemple: NomClasse.hpp/NomClasse.h, NomClasse.cpp,ouClassName.tpp.Ensuite, si vous avez un fichier d'en-tête contenant la définition d'une classe "BrickWall" représentant un mur de briques, son nom seraBrickWall.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 appliqué dans les modules C++. Vous pouvez suivre votre favori. 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. Place au C++ ! Donc:

- Vous êtes autorisé à utiliser presque tout de la bibliothèque standard. Ainsi, au lieu de vous en tenir à ce que vous savez déjà, il serait judicieux d'utiliser autant que possible les versions C++-ish des fonctions C auxquelles vous êtes habitué.
- Cependant, vous ne pouvez utiliser aucune autre bibliothèque externe. Cela signifie C++11 (et formes dérivées) etAugmenterles bibliothèques sont interdites. Les fonctions suivantes sont également interdites : *printf(), *alloc()etgratuit().Si vous les utilisez, votre note sera de 0 et c'est tout.

- Notez que, sauf indication contraire explicite, leen utilisant l'espace de noms <ns_name>et amiles mots clés sont interdits. Sinon, votre note sera -42.
- **Vous êtes autorisé à utiliser la STL dans les modules 08 et 09 uniquement.**Cela veut dire : non**Conteneurs**(vecteur/liste/carte/et ainsi de suite) et non**Algorithmes**(tout ce qui nécessite d'inclure le <algorithme>en-tête) jusque-là. Sinon, votre note sera -42.

Quelques exigences de conception

- Les fuites de mémoire se produisent également en C++. Lorsque vous allouez de la mémoire (en utilisant le nouveau mot-clé), vous devez éviter**fuites de mémoire**.
- Du module 02 au module 09, vos cours doivent être conçus dans le**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 à l'exercice.
- Vous devriez pouvoir utiliser chacun de vos en-têtes indépendamment des autres. Ainsi, ils doivent inclure toutes les dépendances dont ils ont besoin. Cependant, vous devez éviter le problème de la double inclusion en ajoutant**inclure des gardes**. Sinon, votre note sera de 0.

Lis-moi

- Vous pouvez ajouter des fichiers supplémentaires si vous en avez besoin (c'est-à-dire pour diviser votre code). Comme ces affectations ne sont pas vérifiées par un programme, n'hésitez pas à le faire tant que vous remettez 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 complètement chaque module avant de commencer ! Vraiment, fais-le.
- Par Odin, par Thor ! Utilise ton cerveau!!!

Vous devrez implémenter beaucoup de classes. Cela peut sembler fastidieux, à moins que vous ne puissiez créer un script avec votre éditeur de texte préféré.

Vous disposez d'une certaine liberté pour effectuer les exercices. Cependant, suivez les règles obligatoires et ne soyez pas paresseux. Vous manqueriez beaucoup d'informations utiles ! N'hésitez pas à vous renseigner sur les concepts théoriques.

Chapitre III

Règles spécifiques au module

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

Une fois qu'un conteneur est utilisé, vous ne pouvez pas 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 contenant pour chaque exercice à l'exception de l'exercice 02 qui nécessite l'utilisation de deux contenants.

Vous devez soumettre unMakefilepour chaque programme qui compilera vos fichiers sources vers la sortie requise avec les drapeaux -Mur, -Wextraet -Erreur.

Vous devez utiliser c++ et votre Makefile ne doit pas être réassocié.

TonMakefiledoit au moins contenir les règles $(NOM), tous, nettoyer, fcleanetconcernant.

Chapitre IV

Exercice 00 : Échange Bitcoin

	Exercice : 00
Échange Bitcoin	
Répertoire de remise :*ex*00/	
Fichiers à rendre :Makefile, main.cpp, BitcoinExchange.{cpp, hpp}	
Fonctions interdites :Aucun	

Vous devez créer un programme qui affiche la valeur d'une certaine quantité 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 prendra en entrée une deuxième base de données, stockant 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 sera toujours au format suivant : Année-Mois-Jour.
- Une valeur valide doit être soit un flottant, soit un entier positif compris entre 0 et 1000.

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

Voici un exemple de fichier input.txt :

```
$> head input.txt
rendez-vous | 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 et la date associée.

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

Voici un exemple d'utilisation du programme.

```
$> ./btc
Erreur : impossible d'ouvrir le fichier.
$> ./btc input.txt
03/01/2011 => 3 = 0,9
03/01/2011 => 2 = 0,6
03/01/2011 => 1 = 0,3
03/01/2011 => 1,2 = 0,36
09/01/2011 => 1 = 0,32
Erreur : pas un nombre positif.
Erreur : mauvaise saisie => 2001-42-42
11/01/2012 => 1 = 7.1
Erreur : nombre trop grand.
$>
```

Attention : Le(s) conteneur(s) que vous utilisez pour valider cet exercice ne seront plus utilisables pour la suite de ce module.

Chapitre V

Exercice 01 : Notation polonaise inversée

	Exercice : 01
RPN	
Répertoire de remise :*ex*01/	
Fichiers à rendre :Makefile, main.cpp, RPN.{cpp, hpp}	
Fonctions interdites :Aucun	

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 seront toujours inférieurs à 10.
- 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 pouvoir gérer les 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 + 1)"
Erreur
$>
```

Attention : Le(s) contenant(s) que vous avez utilisé dans l'exercice précédent sont interdits ici. Le ou les contenants que vous avez utilisés pour valider cet exercice ne seront pas utilisables pour la suite de ce module.

Chapitre VI

Exercice 02 : PmergeMe

	Exercice : 02
PmergeMe	
Répertoire de remise :*ex*02/	
Fichiers à rendre :Makefile, main.cpp, PmergeMe.{cpp, hpp}	
Fonctions interdites :Aucun	

Vous devez créer un programme avec ces contraintes :

- Le nom du programme est PmergeMe.
- Votre programme doit être capable d'utiliser une suite d'entiers positifs comme argument.
- Votre programme doit utiliser un algorithme de tri par fusion-insertion pour trier la séquence d'entiers positifs.
- Si une erreur survient 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 nombres entiers différents.

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

Voici quelques directives supplémentaires sur les informations que vous devez 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 précisant le premier conteneur utilisé pour trier la suite 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 suite d'entiers positifs.

Le format d'affichage du temps utilisé pour effectuer votre tri est libre mais la précision choisie doit permettre de bien voir la différence entre les deux contenants 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 gamme de      5 éléments avec std::[..] : 0.00031 us
Temps de traitement d'une gamme 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
$>
```

Avertissement : Le(s) conteneur(s) que vous avez utilisé dans les exercices précédents sont interdit ici.

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

Chapitre VII

Soumission et évaluation par les pairs

Remettez votre devoir dans votreGitedépôt comme d'habitude. Seul le travail à l'intérieur de votre référentiel 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.