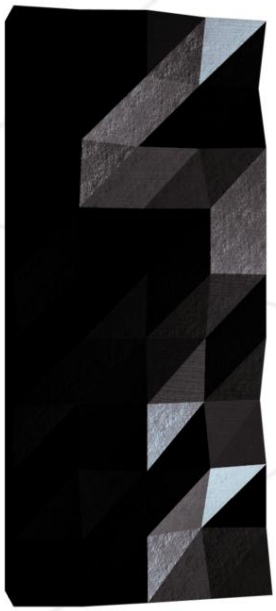

C++ - Module 09

STL

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

Version : 3.0

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 avec classes » (source : Wikipédia).

L'objectif de ces modules est de vous initier à la programmation orientée objet.
Ce sera le point de départ de votre apprentissage du C++. De nombreux langages sont recommandés pour apprendre la POO. Nous avons choisi le C++, car il est dérivé du C.
Parce qu'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 souhaitez devenir un développeur C++ compétent, il vous appartient de poursuivre vos études après le Common Core 42 !

Chapitre II

Règles générales

Compilation

- Compilez votre code avec c++ et les indicateurs -Wall -Wextra -Werror
- Votre code devrait toujours être compilé si vous ajoutez l'indicateur -std=c++98

Conventions de formatage et de dénomination

- Les répertoires d'exercices seront nommés de cette façon : ex00, ex01, ... , exn
- Nommez vos fichiers, classes, fonctions, fonctions membres et attributs comme requis dans les lignes directrices.
- Écrivez les noms de classe en majuscules (UpperCamelCase) . Les fichiers contenant du code de classe seront toujours nommés selon le nom de la classe. Par exemple : ClassName.hpp/ClassName.h, ClassName.cpp ou ClassName.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, chaque message de sortie doit se terminer par un caractère de nouvelle ligne et être affiché sur la sortie standard.
- Adieu Norminette ! Aucun style de codage n'est imposé dans les modules C++. Vous pouvez suivre votre style préféré. Mais gardez à l'esprit que le code que vos pairs ne comprennent pas est du code qu'ils ne peuvent pas noter. Faites de votre mieux pour écrire un code clair et lisible.

Autorisé/Interdit

Vous ne codez plus en C. Passez au C++ ! Par conséquent :

- Vous pouvez utiliser presque tout ce qui est proposé dans la bibliothèque standard. Ainsi, plutôt que 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 utiliser aucune autre bibliothèque externe. Cela signifie que C++11 (et ses dérivés) et les bibliothèques Boost sont interdits. Les fonctions suivantes sont également interdites : *printf(), *alloc() et free(). Si vous les utilisez, votre note sera de 0, point final.

- Notez que, sauf indication contraire explicite, l'espace de noms d'utilisation <ns_name> et Les mots-clés amis sont interdits. Sinon, votre note sera de -42.
- Vous êtes autorisé à utiliser le STL uniquement dans les modules 08 et 09. Cela signifie : pas de conteneurs (vecteur/liste/carte, etc.) ni d'algorithmes (tout ce qui nécessite l'inclusion de l'en-tête <algorithm>) jusqu'à cette date. Sinon, votre note sera de -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 les fuites de mémoire.
- Du module 02 au module 09, vos cours doivent être conçus dans le style orthodoxe Forme canonique, 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 nécessaires. Cependant, vous devez éviter le problème de double inclusion en ajoutant des gardes d'inclusion. Sinon, votre note sera de 0.

Lis-moi

- Vous pouvez ajouter des fichiers supplémentaires si nécessaire (par exemple, pour fractionner votre code). Ces devoirs n'étant pas vérifiés par un programme, n'hésitez pas à le faire, à condition de fournir les fichiers obligatoires.
- Parfois, les lignes directrices d'un exercice semblent courtes, mais les exemples peuvent montrer exigences qui ne sont pas explicitement écrites dans les instructions.
- Lisez chaque module en entier avant de commencer ! Vraiment, foncez.
- Par Odin, par Thor ! Utilisez votre cerveau !!!

Concernant le Makefile pour les projets C++, les mêmes règles qu'en C s'appliquent (voir le chapitre Norme sur le Makefile).

Vous devrez implémenter de nombreuses classes. Cela peut paraître fastidieux, à moins que vous ne puissiez écrire des scripts dans votre éditeur de texte préféré.

Vous disposez d'une certaine liberté pour réaliser les exercices.
Cependant, suivez les règles obligatoires et ne soyez pas paresseux. Vous
Vous manquez beaucoup d'informations utiles ! N'hésitez pas à lire
concepts théoriques.

Chapitre III

Règles spécifiques au module

Il est obligatoire d'utiliser les conteneurs standards 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 le exercices.

Vous devez utiliser au moins un récipient pour chaque exercice avec le exception de l'exercice 02 qui nécessite l'utilisation de deux conteneurs.

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

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

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

Chapitre IV

Instructions de l'IA

• Context

Ce projet est conçu pour vous aider à découvrir les éléments fondamentaux de votre formation en TIC.

Pour ancrer correctement les connaissances et les compétences clés, il est essentiel d'adopter une approche réfléchie de l'utilisation des outils et du support de l'IA.

Un véritable apprentissage fondamental nécessite un véritable effort intellectuel — par le biais de défis, de répétitions et d'échanges d'apprentissage entre pairs.

Pour un aperçu plus complet de notre position sur l'IA — en tant qu'outil d'apprentissage, en tant qu'élément du programme des TIC et en tant qu'attente sur le marché du travail — veuillez vous référer à la FAQ dédiée sur l'intranet.

• Message principal

Construisez des bases solides sans raccourcis.

Développez réellement vos compétences techniques et énergétiques.

Vivez un véritable apprentissage entre pairs, commencez à apprendre à apprendre et à résoudre de nouveaux problèmes.

Le parcours d'apprentissage est plus important que le résultat.

Apprenez-en davantage sur les risques associés à l'IA et développez des pratiques de contrôle efficaces et des contre-mesures pour éviter les pièges courants.

• Règles de l'apprenant :

• Vous devez appliquer le raisonnement aux tâches qui vous sont assignées, en particulier avant de vous tourner vers l'IA.

- Vous ne devez pas demander de réponses directes à l'IA.
- Vous devriez en apprendre davantage sur l'approche globale 42 de l'IA.

• Résultats de la phase :

Au cours de cette phase fondamentale, vous obtiendrez les résultats suivants :

- Obtenez des bases techniques et de codage appropriées.
- Sachez pourquoi et comment l'IA peut être dangereuse durant cette phase.

• Commentaires et exemple :

- Oui, nous savons que l'IA existe et qu'elle peut résoudre vos projets. Mais vous êtes ici pour apprendre, pas pour prouver que l'IA a appris. Ne perdez pas votre temps (ni le nôtre) à simplement démontrer que l'IA peut résoudre un problème donné.

Apprendre à 42 ans ne consiste pas à connaître la réponse, mais à développer la capacité d'en trouver une. L'IA vous donne directement la réponse, mais cela vous empêche de construire votre propre raisonnement. Et le raisonnement demande du temps, des efforts et implique des échecs. Le chemin vers le succès n'est pas censé être facile.

- Gardez à l'esprit que pendant les examens, l'IA n'est pas disponible : pas d'Internet, pas de smartphones, etc. Vous vous rendrez vite compte si vous vous êtes trop appuyé sur l'IA dans votre processus d'apprentissage.
- L'apprentissage entre pairs vous expose à des idées et des approches différentes, améliorant ainsi vos compétences interpersonnelles et votre capacité à penser différemment. C'est bien plus précieux que de simplement discuter avec un robot. Alors n'hésitez pas : discutez, posez des questions et apprenez ensemble !
- Oui, l'IA sera intégrée au programme, à la fois comme outil d'apprentissage et comme sujet à part entière. Vous aurez même la possibilité de créer votre propre logiciel d'IA. Pour en savoir plus sur notre approche crescendo, vous consulterez la documentation disponible sur l'intranet.

Bonnes pratiques :

> Je suis bloqué sur un nouveau concept. Je demande à quelqu'un à proximité comment il l'a abordé. On discute 10 minutes, et soudain, ça fait tilt. Je comprends.

Mauvaise pratique :

> J'utilise l'IA en secret, je copie du code qui semble correct. Lors de l'évaluation par les pairs, je ne peux rien expliquer. J'échoue. À l'examen, sans IA, je suis à nouveau bloqué. J'échoue.

Chapitre V

Exercice 00 : Échange de Bitcoins

	Exercice : 00
Bourse Bitcoin	
Répertoire de remise :	
ex00/ Fichiers à remettre : Makefile, main.cpp, BitcoinExchange.{cpp, hpp}	
Fonctions interdites : Aucune	

Vous devez créer un programme qui génère 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 au fil du 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 comme argument.
- Chaque ligne de ce fichier doit utiliser le format suivant : « date | valeur ».
- Une date valide sera toujours au format suivant : Année-Mois-Jour.
- Une valeur valide doit être soit un nombre flottant, soit un entier positif, compris entre 0 et 1 000.

Vous devez utiliser au moins un conteneur dans votre code pour valider cet exercice. Gérez les erreurs potentielles avec une méthode appropriée.
message d'erreur.

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 selon la date indiquée dans votre base de données.

Si la date utilisée en entrée n'existe pas dans votre base de données, vous devez utiliser la date la plus proche. Veillez à utiliser la date la plus basse et non la plus haute.

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 : ce n'est 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) que vous utilisez pour valider cet exercice ne seront plus utilisables pour la suite de ce module.

Chapitre VI

Exercice 01 : Notation polonaise inversée

	Exercice : 01
	Infirmière auxiliaire autorisée
Répertoire de remise : ex01/	
Fichiers à remettre : 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.
ment.

• Les nombres utilisés dans cette opération et passés en arguments seront toujours inférieurs à 10. Le
calcul lui-même mais aussi le résultat ne tiennent pas compte de cette règle.

• Votre programme doit traiter cette expression et afficher le résultat correct sur le
sortie standard.

• Si une erreur survient pendant l'exécution du programme, un message d'erreur doit être affiché
affiché sur l'erreur standard.

• Votre programme doit être capable de gérer les opérations avec ces jetons : « + - / * ».

Vous devez utiliser au moins un conteneur dans votre code pour valider cela
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 - 4 - 1 +" 42 $> ./RPN "7 7 * 7 -" 42 $> ./RPN
"1
2 * 2 / 2 * 2 4 - +" 0 $> ./RPN "(1 + 1)"



Erreur $>
```

Attention : Les conteneurs utilisés dans l'exercice précédent sont interdits ici. Les conteneurs utilisés pour valider cet exercice
ne sera pas utilisable pour le reste de ce module.

Chapitre VII

Exercice 02 : PmergeMe

	Exercice : 02
PmergeMe	
Répertoire de remise :	
ex02/ Fichiers à remettre : 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 l'entier positif séquence.

Pour clarifier, oui, vous devez utiliser l'algorithme de Ford-Johnson. (source : Art Of Computer Programming, Vol.3. (Insertion de fusion, page 184.)

- Si une erreur se produit pendant l'exécution du programme, un message d'erreur doit s'afficher sur l'erreur standard.

Vous devez utiliser au moins deux conteneurs différents dans votre code pour valider cet exercice. Votre programme doit être capable de gérer au moins 3 000 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 du positif non trié séquence d'entiers.
- Sur la deuxième ligne, vous devez afficher un texte explicite suivi du positif trié séquence d'entiers.
- Sur la troisième ligne, vous devez afficher un message explicite indiquant le temps pris par votre algorithme, en précisant le premier conteneur utilisé pour trier l'entier positif séquence.
- 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 le 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 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 l'utilisateur OSX : $> ./
PmergeMe `jot -r 3000 1 100000 | tr '\n' ' '` [...] $>
```

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

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

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

Chapitre VIII

Soumission et évaluation par les pairs

Remettez votre devoir dans votre dépôt Git comme d'habitude. Seul le travail effectué dans vos dépôts 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.

Lors de l'évaluation, une brève modification du projet peut être occasionnellement demandée. Il peut s'agir d'un changement de comportement mineur, de quelques lignes de code à écrire ou à réécrire, ou d'une fonctionnalité facile à ajouter.

Bien que cette étape ne soit pas applicable à tous les projets, vous devez vous y préparer si elle est mentionnée dans les directives d'évaluation.

Cette étape vise à vérifier votre compréhension réelle d'une partie spécifique du projet.
La modification peut être effectuée dans n'importe quel environnement de développement de votre choix (par exemple, votre configuration habituelle) et devrait être réalisable en quelques minutes, à moins qu'un délai spécifique ne soit défini dans le cadre de l'évaluation.
On peut par exemple vous demander d'effectuer une petite mise à jour d'une fonction ou d'un script, de modifier un affichage ou d'ajuster une structure de données pour stocker de nouvelles informations, etc.

Les détails (portée, cible, etc.) seront précisés dans les lignes directrices d'évaluation et peuvent varier d'une évaluation à l'autre pour un même projet.

16D85ACC441674FBA2DF65190663F33F793984B142405F56715D5225FBAB6E3D6A4F
167020A16827E1B16612137E59ECD492E47AB764CB10B45D979615AC9FC74D521D9
20A778A5E