



C++ - Module 00

Namespaces, classes, fonctions membres, stdio
streams,
listes d'initialisation, static, const, et autres bases

Résumé:

Ce document contient les exercices du Module 00 des C++ modules.

Version: 9

Table des matières

I	Introduction	2
II	Consignes générales	3
III	Exercice 00 : Mégaphone	5
IV	Exercice 01 : My Awesome PhoneBook	6
V	Exercice 02 : L'emploi de vos rêves	8
VI	Submission and peer-evaluation	10

Chapitre I

Introduction

C++ is a general-purpose programming language created by Bjarne Stroustrup as an extension of the C programming language, or "C with Classes" (source : Wikipedia).

C++ est un langage de programmation compilé permettant la programmation sous de multiples paradigmes, dont la programmation procédurale, la programmation orientée objet et la programmation générique. Ses bonnes performances, et sa compatibilité avec le C en font un des langages de programmation les plus utilisés dans les applications où la performance est critique (source : Wikipedia).

Ces modules ont pour but de vous introduire à la **Programmation Orientée Objet**. Plusieurs langages sont recommandés pour l'apprentissage de l'OOP. Du fait qu'il soit dérivé de votre bon vieil ami le C, nous avons choisi le langage C++. Toutefois, étant un langage complexe et afin de ne pas vous compliquer la tâche, vous vous conformerez au standard C++98.

Nous avons conscience que le C++ moderne est différent sur bien des aspects. Si vous souhaitez pousser votre maîtrise du C++, c'est à vous de creuser après le tronc commun de 42 !

Vous découvrirez de nouveaux concepts au fur et à mesure. Leur difficulté ira croissante.

Chapitre II

Consignes générales

Compilation

- Compilez votre code avec `c++` et les flags `-Wall -Wextra -Werror`
- Votre code doit compiler si vous ajoutez le flag `-std=c++98`

Format et conventions de nommage

- Les dossiers des exercices seront nommés ainsi : `ex00, ex01, ..., exn`
- Nommez vos fichiers, vos classes, vos fonctions, vos fonctions membres et vos attributs comme spécifié dans les consignes.
- Rédigez vos noms de classe au format **UpperCamelCase**. Les fichiers contenant le code d'une classe porteront le nom de cette dernière. Par exemple : `NomDeClasse.hpp/NomDeClasse.h, NomDeClasse.cpp, ou NomDeClasse.tpp`. Ainsi, si un fichier d'en-tête contient la définition d'une classe "BrickWall", son nom sera `BrickWall.hpp`.
- Sauf si spécifié autrement, tous les messages doivent être terminés par un retour à la ligne et être affichés sur la sortie standard.
- *Ciao Norminette !* Aucune norme n'est imposée durant les modules C++. Vous pouvez suivre le style de votre choix. Mais ayez à l'esprit qu'un code que vos pairs ne peuvent comprendre est un code que vos pairs ne peuvent évaluer. Faites donc de votre mieux pour produire un code propre et lisible.

Ce qui est autorisé et ce qui ne l'est pas

Le langage C, c'est fini pour l'instant. Voici l'heure de se mettre au C++ ! Par conséquent :

- Vous pouvez avoir recours à quasi l'ensemble de la bibliothèque standard. Donc plutôt que de rester en terrain connu, essayez d'utiliser le plus possible les versions C++ des fonctions C dont vous avez l'habitude.
- Cependant, vous ne pouvez avoir recours à aucune autre bibliothèque externe. Ce qui signifie que C++11 (et dérivés) et l'ensemble **Boost** sont interdits. Aussi, certaines fonctions demeurent interdites. Utiliser les fonctions suivantes résultera

en la note de 0 : `*printf()`, `*alloc()` et `free()`.

- Sauf si explicitement indiqué autrement, les mots-clés `using namespace <ns_name>` et `friend` sont interdits. Leur usage résultera en la note de -42.
- **Vous n'avez le droit à la STL que dans les Modules 08 et 09.** D'ici là, l'usage des **Containers** (`vector/list/map/etc.`) et des **Algorithmes** (tout ce qui requiert d'inclure `<algorithm>`) est interdit. Dans le cas contraire, vous obtiendrez la note de -42.

Quelques obligations côté conception

- Les fuites de mémoires existent aussi en C++. Quand vous allouez de la mémoire (en utilisant le mot-clé `new`), vous ne **devez pas avoir de memory leaks**.
- Du Module 02 au Module 09, vos classes devront se conformer à la forme **canonique, dite de Coplien, sauf si explicitement spécifié autrement**.
- Une fonction implementée dans un fichier d'en-tête (hormis dans le cas de fonction template) équivaudra à la note de 0.
- Vous devez pouvoir utiliser vos fichiers d'en-tête séparément les uns des autres. C'est pourquoi ils devront inclure toutes les dépendances qui leur seront nécessaires. Cependant, vous devez éviter le problème de la double inclusion en les protégeant avec des **include guards**. Dans le cas contraire, votre note sera de 0.

Read me

- Si vous en avez le besoin, vous pouvez rendre des fichiers supplémentaires (par exemple pour séparer votre code en plus de fichiers). Vu que votre travail ne sera pas évalué par un programme, faites ce qui vous semble le mieux du moment que vous rendez les fichiers obligatoires.
- Les consignes d'un exercice peuvent avoir l'air simple mais les exemples contiennent parfois des indications supplémentaires qui ne sont pas explicitement demandées.
- Lisez entièrement chaque module avant de commencer ! Vraiment.
- Par Odin, par Thor ! Utilisez votre cervelle !!!



Vous aurez à implémenter un bon nombre de classes, ce qui pourrait s'avérer ardu... ou pas ! Il y a peut-être moyen de vous simplifier la vie grâce à votre éditeur de texte préféré.



Vous êtes assez libre quant à la manière de résoudre les exercices. Toutefois, respectez les consignes et ne vous en tenez pas au strict minimum, vous pourriez passer à côté de notions intéressantes. N'hésitez pas à lire un peu de théorie.

Chapitre III

Exercice 00 : Mégaphone

	Exercice : 00
	Mégaphone
	Dossier de rendu : <i>ex00/</i>
	Fichiers à rendre : Makefile, megaphone.cpp
	Fonctions interdites : Aucune

Afin de s'assurer que tout le monde est bien réveillé, écrivez un programme qui se comporte comme suit :

```
$>./megaphone "shhhh... I think the students are asleep..."  
SHHHHHH... I THINK THE STUDENTS ARE ASLEEP...  
$>./megaphone Damnit " ! " "Sorry students, I thought this thing was off."  
DAMNIT ! SORRY STUDENTS, I THOUGHT THIS THING WAS OFF.  
$>./megaphone  
* LOUD AND UNBEARABLE FEEDBACK NOISE *  
$>
```



Faites les exercices d'une "façon" C++.

Chapitre IV

Exercice 01 : My Awesome PhoneBook

	Exercice : 01
	My Awesome PhoneBook
Dossier de rendu : <i>ex01/</i>	
Fichiers à rendre : Makefile , *.cpp, *.{h, hpp}	
Fonctions interdites : Aucune	

Bienvenue dans les années 80 et leur incroyable technologie! Concevez un programme qui se comporte comme un répertoire pas si incroyable que ça.

Vous devez implémenter deux classes :

- **PhoneBook**
 - Représente le répertoire.
 - Contient un tableau de contacts.
 - Peut enregistrer **8 contacts** maximum. Si l'utilisateur tente d'ajouter un 9ème contact, remplacez le plus ancien par celui-ci.
 - Notez que l'allocation dynamique est interdite.
- **Contact**
 - Représente un contact dans le répertoire.

Dans votre code, l'objet répertoire doit être une instance de la classe **PhoneBook**. Même chose pour les objets contacts, qui doivent être chacun une instance de la classe **Contact**. Vous êtes libre de concevoir vos classes comme vous le sentez. Par contre, ayez en tête que ce qui est toujours utilisé dans une classe est privé, et que ce qui peut l'être en dehors est public.



N'oubliez pas de regarder les vidéos de l'intranet.

Au lancement du programme, le répertoire est vide et l'utilisateur peut entrer une commande. Le programme accepte les entrées suivantes : ADD, SEARCH et EXIT.

- **ADD** : enregistrer un nouveau contact
 - Si l'utilisateur entre cette commande, le programme lui demande de remplir une par une les informations du nouveau contact. Une fois tous les champs complétés, le nouveau contact est ajouté au répertoire.
 - Un contact possède les champs suivants : first name (*prénom*), last name (*nom de famille*), nickname (*surnom*), phone number (*numéro de téléphone*), et darkest secret (*son plus lourd secret*). Les champs d'un contact enregistré ne peuvent être vides.
- **SEARCH** : affiche le contact demandé
 - Affiche les contacts enregistrés sous la forme d'une liste de **4 colonnes** : index, first name, last name et nickname.
 - Chaque colonne doit faire **10 caractères** de long. Elles doivent être séparées par un pipe ('|'). Leur texte est aligné à droite. Si le texte dépasse la largeur de la colonne, il faut le tronquer et remplacer le dernier caractère affiché par un point ('.').
 - Ensuite, le programme demande à l'utilisateur d'entrer l'index du contact à afficher. Si l'index ou son format sont incorrects, gérez cela de manière pertinente. Sinon, affichez les informations du contact, une par ligne.
- **EXIT**
 - Le programme quitte et les contacts sont perdus à jamais !
- **Toute autre entrée est ignorée.**

Une fois la commande correctement exécutée, le programme attend à nouveau une entrée. Il prend fin lorsque l'utilisateur entre EXIT.

Donnez un nom cohérent à votre exécutable.



<http://www.cplusplus.com/reference/string/string/> et bien entendu
<http://www.cplusplus.com/reference/iomanip/>

Chapitre V

Exercice 02 : L'emploi de vos rêves

	Exercice : 02
	L'emploi de vos rêves
Dossier de rendu :	<i>ex02/</i>
Fichiers à rendre :	Makefile , Account.cpp , Account.hpp , tests.cpp
Fonctions interdites :	Aucune



Account.hpp, tests.cpp et le journal sont à télécharger via la page du module dans l'intranet.

C'est votre premier jour à *GlobalBanksters United*. Après avoir réussi les différentes phases de recrutement haut la main (merci à cet ami qui a eu la bonne idée de vous montrer quelques astuces *Microsoft Office*), vous avez rejoint l'équipe de dev. Vous avez conscience que votre installation éclair d'*Adobe Reader* a fait forte impression. Ce petit plus a fait toute la différence et vous a permis de vaincre vos ennemis (oui, les autres candidats)! Vous avez réussi!

C'est pas tout mais votre responsable vient de vous confier du travail. Votre première mission : recréer un fichier manquant. Quelque chose a mal tourné et un fichier source a été supprimé par erreur. Malheureusement, vos collègues ne connaissent pas **Git** et se servent de clés USB pour se partager le code. Il serait sensé de partir de cet endroit dès maintenant. Malgré tout, vous décidez de rester et de relever le défi.

On vous transmet quelques fichiers. Compiler **tests.cpp** vous confirme que le fichier manquant est **Account.cpp**. Heureusement, le fichier d'en-tête **Account.hpp** est indemne, ainsi qu'un journal (fichier de log). Peut-être pourriez-vous utiliser ce dernier afin de comprendre comment était implémentée la classe **Account** (*compte bancaire*).

Vous commencez à refaire le fichier **Account.cpp**. En quelques minutes à peine, vous

avez tapé des lignes de C++ de génie. Après quelques compilations ratées, votre programme passe les tests. Sa sortie correspond à la perfection à celle contenue dans le journal (**sauf pour l'horodatage**, qui sera forcément différent étant donné que les tests sauvegardés dans le journal ont été faits bien avant votre arrivée).

Bon sang, mais quel talent !



L'ordre dans lequel les destructeurs sont appelés peut différer selon votre compilateur/système d'exploitation. Ainsi, vos destructeurs peuvent être appelés dans l'ordre inverse.



Vous pouvez valider ce module sans l'exercice 02.

Chapitre VI

Submission and peer-evaluation

Rendez votre travail sur votre dépôt **Git** comme d'habitude. Seul le travail présent sur votre dépôt sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.



????????????? XXXXXXXXXX = \$3\$\$f15bc138aca1e76ec6f4cf0797ec037