



C++ - Module 07

Modèles C++

Résumé :

Ce document contient les exercices du module 07 des modules C++.

Version : 7

Contenu

I	Introduction	2
II	Règles générales	3
III	Exercice 00 : Commencer avec quelques fonctions	5
IV	Exercice 01 : Iter	7
V	Exercice 02 : tableau	8
VI	Soumission et évaluation par les pairs	9

Chapitre I

Introduction

Le C++ est un langage de programmation généraliste créé par Bjarne Stroustrup en tant qu'extension 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.hpp. 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'écrire des scripts dans votre éditeur de texte préféré.



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

Exercice 00 : commencer par quelques fonctions

	Exercice : 00
Commencez par quelques fonctions	
Annuaire de retournement : <i>ex00/</i>	
Fichiers à rendre : Makefile, main.cpp, whatever.{h, hpp}	
Fonctions interdites : Aucune	

Mettre en œuvre les modèles de fonction suivants :

- swap : Échange les valeurs de deux arguments donnés. Ne renvoie rien.
- min : compare les deux valeurs passées en argument et renvoie la plus petite. Si les deux sont égales, il renvoie la seconde.
- max : Compare les deux valeurs passées en argument et renvoie la plus grande. Si les deux sont égales, il renvoie la seconde.

Ces fonctions peuvent être appelées avec n'importe quel type d'argument. La seule condition est que les deux arguments soient du même type et qu'ils supportent tous les opérateurs de comparaison.



Les modèles doivent être définis dans les fichiers d'en-tête.

Exécuter le code suivant :


```
int    main( void ) {  
  
    int a = 2 ;  
    int b = 3 ;  
  
    ::swap( a, b ) ;  
    std::cout << "a = " << a << ", b = " << b << std::endl ;  
    std::cout << "min( a, b ) = " << ::min( a, b ) << std::endl ;  
    std::cout << "max( a, b ) = " << ::max( a, b ) << std::endl ;  
  
    std::string c = "chaine1" ;  
    std::string d = "chaine2" ;  
  
    ::swap( c, d ) ;  
    std::cout << "c = " << c << ", d = " << d << std::endl ;  
    std::cout << "min( c, d ) = " << ::min( c, d ) << std::endl ;  
    std::cout << "max( c, d ) = " << ::max( c, d ) << std::endl ;  
  
    retour 0 ;  
}
```

Devrait sortir :

```
a = 3, b = 2  
min(a, b) = 2  
max(a, b) = 3  
c = chaine2, d = chaine1  
min(c, d) = chaine1  
max(c, d) = chaine2
```

Chapitre IV

Exercice 01 : Iter

	Exercice : 01
Iter	
Répertoire de	
retournement : Makefile, main.cpp, iter.{h,	
hpp}	

Fonctions interdites : Aucune

Implémenter une fonction template iter qui prend 3 paramètres et ne renvoie rien.

- Le premier paramètre est l'adresse d'un tableau.
- La seconde est la longueur du tableau.
- La troisième est une fonction qui sera appelée sur chaque élément du tableau.


Remettez un fichier main.cpp contenant vos tests. Fournissez suffisamment de code pour générer un exécutable de test.

Votre modèle de fonction iter doit fonctionner avec n'importe quel type de tableau. Le troisième paramètre peut être un modèle de fonction instancié.

Chapitre V

Exercice 02 :

tableau

	Exercice : 02
Tableau	
Annuaire de retournement : <i>ex02/</i>	
Fichiers à rendre : Makefile, main.cpp, Array.{h, hpp} et fichier optionnel : Array.tpp	
Fonctions interdites : Aucune	

Développez un modèle de classe **Array** qui contient des éléments de type T et qui implémente le comportement et les fonctions suivants :

- Construction sans paramètre : Crée un tableau vide.
- Construction avec un int non signé n comme paramètre : Crée un tableau de n éléments initialisés par défaut.
Conseil : essayez de compiler `int * a = new int()` ; puis d'afficher *a.
- Construction par l'opérateur de copie et d'affectation. Dans les deux cas, la modification du tableau original ou de sa copie après la copie ne doit pas affecter l'autre tableau.
- Vous DEVEZ utiliser l'opérateur `new[]` pour allouer de la mémoire. L'allocation préventive (localisation de la mémoire à l'avance) est interdite. Votre programme ne doit jamais accéder à une mémoire non allouée.
- Les éléments sont accessibles par l'intermédiaire de l'opérateur d'indice : `[]`.
- Lors de l'accès à un élément avec l'opérateur `[]`, si son index est hors limites, un message d'erreur `std::exception` est levée.
- Une fonction membre `size()` qui renvoie le nombre d'éléments du tableau. Cette fonction membre ne prend aucun paramètre et ne doit pas modifier l'instance courante.

Comme d'habitude, assurez-vous que tout fonctionne comme prévu et remettez un

fichier `main.cpp` c o n t e n a n t vos tests.

Chapitre VI

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.



16D85ACC441674FBA2DF65190663F43A243E8FA5424E49143B520D3DF8AF68036E47
114F20A16827E1B16612137E59ECD492E468BC6CD109F65388DC57A58E8942585C8
D193B96732206