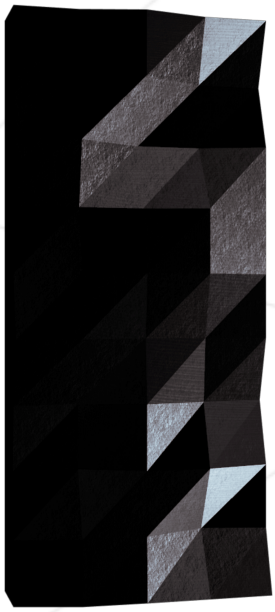


ft_containers

Les containers C++, tout simplement

Résumé:
Les containers en C++ ont tous un usage particulier.
Afin de s'assurer que vous les comprenez, vous allez les implémenter !

Version: 4

Table des matières

Chapitre I

Objectifs

Dans ce projet, vous allez implémenter quelques containers C++ de la bibliothèque standard (*Standard Template Library*).

Vous devez vous baser sur la structure des containers originaux. Si une partie de la forme canonique de Coplien n'y est pas présente, ne la faites pas.

Rappelez-vous. Vous devez vous conformer au standard C++98. Par conséquent, toute fonctionnalité plus récente ne doit **PAS** être réalisée, mais toute fonctionnalité C++98 (même obsolète) est attendue.

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

- Pour chaque container, rendez les fichiers obligatoires avec le nom attendu.
- *Ciao Norminette !* Aucune norme n'est imposée. 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. C'est l'heure du C++ ! Par conséquent :

- Vous pouvez avoir recours à 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 avec 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()`.

Quelques obligations côté conception

- Les fuites de mémoires existent aussi en C++. Quand vous allouez de la mémoire, vous ne **devez pas avoir de memory leaks**.
- Une fonction implémentée 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) et organiser votre rendu comme bon vous semble du moment que vous rendez les fichiers obligatoires.
- Par Odin, par Thor ! Utilisez votre cervelle !!!

Puisque votre but est ici de recoder des containers de la STL, vous ne **pouvez pas utiliser ces derniers** pour votre implémentation.

Chapitre III

Partie obligatoire

Implémentez les containers suivants et rendez les fichiers `<container>.hpp` correspondants :

- `vector`
  Vous n'avez pas à faire la spécialisation `vector<bool>`.
- `map`
- `stack`
  Elle utilisera votre classe `vector` comme container sous-jacent par défaut. Cependant, elle restera compatible avec les autres containers, ceux de la STL inclus.

`Vous pouvez valider ce projet sans la stack (80/100).`
`Cependant, si vous voulez faire la partie bonus, il faudra avoir réalisé les 3 containers obligatoires.`

Vous devez aussi implémenter :

- `iterators_traits`
- `reverse_iterator`
- `enable_if`
  Oui, il s'agit de C++11 mais vous saurez le refaire en C++98.
  Le but est de vous faire découvrir SFINAE.
- `is_integral`
- `equal` et/ou `lexicographical_compare`
- `std::pair`
- `std::make_pair`

III.1 Prérequis

- Le namespace doit être `ft`.
- La structure de données interne utilisée pour chacun de vos containers doit être cohérente et justifiable (utiliser juste un tableau pour `map` n'est donc pas accepté).
- Vous ne pouvez pas implémenter plus de fonctions publiques que ne possèdent les containers standards. Toute autre fonction supplémentaire doit être privée ou protégée. Chaque fonction et chaque variable publique doit être **justifiée**.
- Toutes les fonctions membres, les fonctions non-membres et les surcharges d'un container sont attendues.
- Vous devez vous conformer au nommage original. Faites attention aux détails.
- Si le container possède un système d'**itérateur**, vous devez l'implémenter.
- Vous devez utiliser `std::allocator`.
- Pour les surcharges non-membres, le mot-clé `friend` est autorisé. Chaque utilisation de `friend` doit être justifiée et sera vérifiée en évaluation.
- Bien entendu, pour implémenter `map::value_compare`, le mot-clé `friend` est autorisé.

Vous pouvez utiliser https://www.cplusplus.com/
et https://cppreference.com/ comme références.

III.2 Test

- Vous devez aussi fournir vos propres tests, au minimum un `main.cpp`, pour votre évaluation. Vous devez pousser plus loin que le `main` donné en exemple !
- Vous devez créer deux binaires faisant tourner les mêmes tests : l'un avec vos containers et l'autre avec les containers standards.
- Comparez les **sorties** et les **performances / temps** (vos containers peuvent être jusqu'à 20 fois plus lents que les originaux).
- Pour tester vos containers : `ft::<container>`.

Un fichier main.cpp est disponible sur la page intranet du projet.

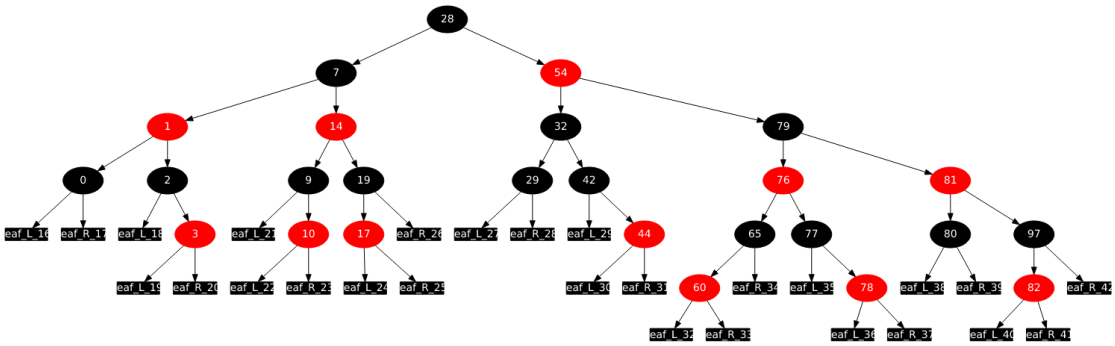
Chapitre IV

Partie bonus

Vous aurez des points bonus si vous implémentez un dernier container :

- `set`

Mais cette fois, un **arbre rouge et noir** est **obligatoire**.



Les bonus ne seront évalués que si la partie obligatoire est PARFAITE. Par parfaite, nous entendons complète et sans aucun dysfonctionnement. Si vous n'avez pas réussi TOUS les points de la partie obligatoire, votre partie bonus ne sera pas prise en compte.

Chapitre V

Rendu et peer-evaluation

Rendez votre travail sur votre dépot `Git` comme d'habitude. Seul le travail présent sur votre dépot 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.