

# More Exceptional C++

par Matthieu Brucher (<http://matthieu-brucher.developpez.com/>) (Blog)

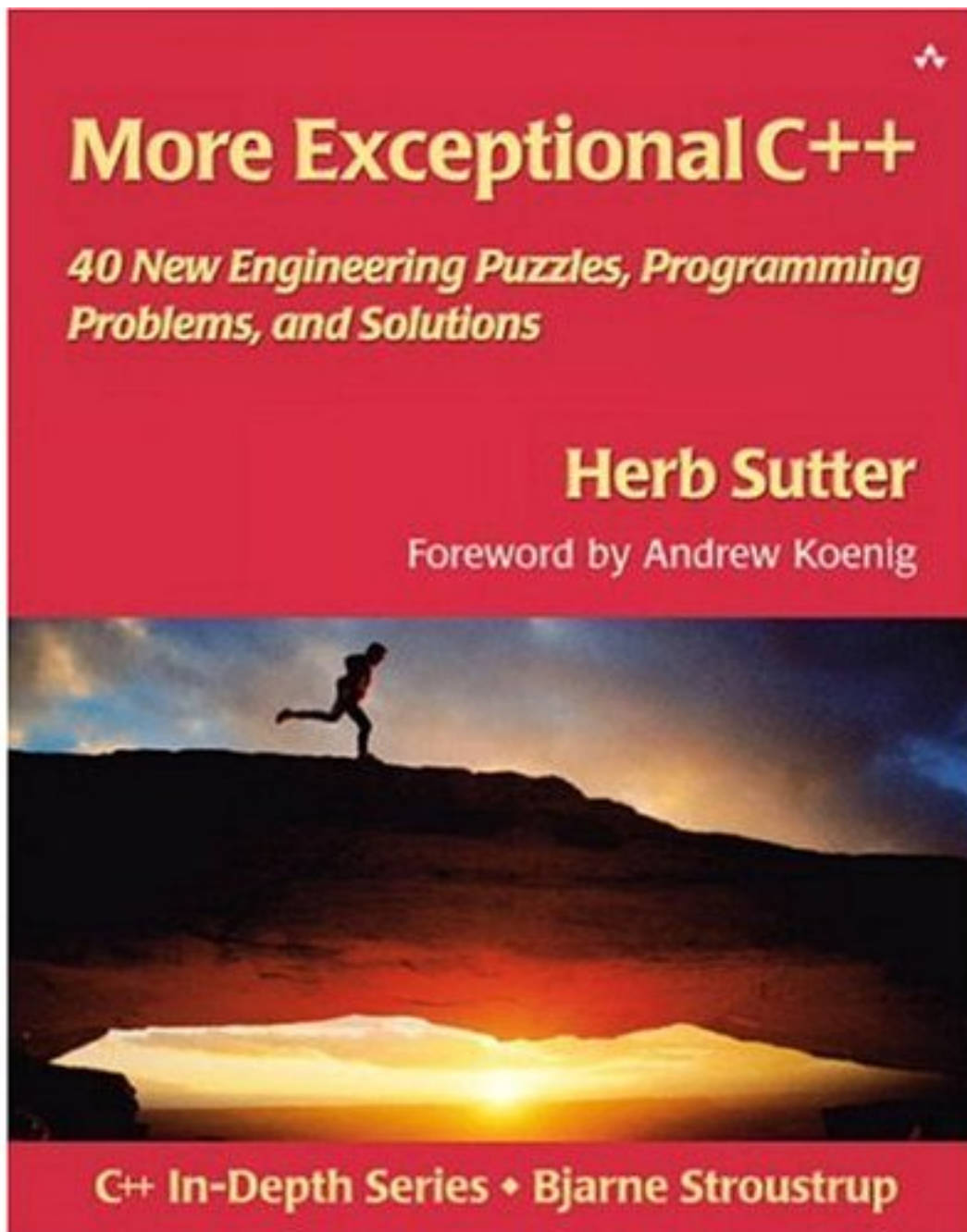
Date de publication : 11/05/2006

Dernière mise à jour : 27/03/2008

Critique de More Exceptional C++ d'*Herb Sutter*

- I - Description
- II - Table des matières
- III - Critique : Moins indispensable que le précédent
- IV - Liens annexes

## I - Description



40 questions pour la suite de Exceptional C++ , toujours d' *Herb Sutter* . Les solutions apportées sont aussi variées et importantes comme :

- Quels sont les dangers de l'utilisation des `std::set` ou `std::map` et comment les éviter ?
- Quelles sont les fonctions dépréciées utilisables avec la STL et celles non utilisables ?
- Quelles sont les techniques disponibles pour générer du code templatisé puissant dont le comportement peut être modifié par le type de données utilisé ?
- Quand et comment optimiser son code ? Quels sont les optimisations pouvant causer des problèmes, surtout en environnement multi-threadé ?

- Comment la gestion des exceptions affecte-elle le design des classes ?
- Comment gérer les problèmes d'héritage avec des bibliothèques de concepteurs différents - problème des soeurs siamoises - ?
- Comment utiliser les `std::auto_ptr` de manière sûre et comment l'adapter avec des patterns simples pour éviter les écueils courants ? Est-il utilisable en tant que membre de classe ? A quel prix ?
- Quelques questions courantes en C++ comme l'utilisation des namespaces : comment et pourquoi ?

## II - Table des matières

- Generic Programming and the C++ Standard Library.
  - Item 1: Switching Streams
  - Item 2: Predicates, Part 1: What remove() Removes
  - Item 3: Predicates, Part 2: Matters of State
  - Item 4: Extensible Templates: Via Inheritance or Traits?
  - Item 5: Typename
  - Item 6: Containers, Pointers, and Containers That Aren't
  - Item 7: Using Vector and Deque
  - Item 8: Using Set and Map
  - Item 9: Equivalent Code?
  - Item 10: Template Specialization and Overloading
  - Item 11: Mastermind
- Optimization and Performance.
  - Item 12: Inline
  - Item 13: Lazy Optimization, Part 1: A Plain Old String
  - Item 14: Lazy Optimization, Part 2: Introducing Laziness
  - Item 15: Lazy Optimization, Part 3: Iterators and References
  - Item 16: Lazy Optimization, Part 4: Multi-Threaded Environments
- Exception Safety Issues and Techniques.
  - Item 17: Constructor Failures, Part 1: Object Lifetimes
  - Item 18: Constructor Failures, Part 2: Absorption?
  - Item 19: Uncaught Exceptions
  - Item 20: An Unmanaged Pointer Problem, Part 1: Parameter Evaluation
  - Item 21: An Unmanaged Pointer Problem, Part 2: What About auto\_ptr?
  - Item 22: Exception-Safe Class Design, Part 1: Copy Assignment
  - Item 23: Exception-Safe Class Design, Part 2: Inheritance
- Inheritance and Polymorphism.
  - Item 24: Why Multiple Inheritance?
  - Item 25: Emulating Multiple Inheritance
  - Item 26: Multiple Inheritance and the Siamese Twin Problem
  - Item 27: (Im)pure Virtual Functions
  - Item 28: Controlled Polymorphism
- Memory and Resource Management.
  - Item 29: Using auto\_ptr
  - Item 30: Smart Pointer Members, Part 1: A Problem with auto\_ptr
  - Item 31: Smart Pointer Members, Part 2: Toward a ValuePtr
- Free Functions and Macros.
  - Item 32: Recursive Declarations
  - Item 33: Simulating Nested Functions
  - Item 34: Preprocessor Macros

- Item 35: #Definition
- Miscellaneous Topics.
  - Item 36: Initialization
  - Item 37: Forward Declarations
  - Item 38: Typedef
  - Item 39: Namespaces, Part 1: Using-Declarations and Using-Directives
  - Item 40: Namespaces, Part 2: Migrating to Namespaces
- Appendixes
  - Appendix A. Optimizations That Aren't (In a Multithreaded World).
  - Appendix B. Test Results for Single- versus Multi-Thread-Safe String Implementations.

### III - Critique : Moins indispensable que le précédent

Moins cohérent que le premier Exceptional C++, ce deuxième opus répond à des questions plus précises. Le seul bloc de questions comparable au premier livre concerne le paradigme COW pour Copy-On-Write, utilisé souvent pour la gestion des strings à moindre coût d'allocation. Plusieurs implémentations des `std::string` utilisent ce paradigme, et des bibliothèques commerciales l'utilisent, comme Qt4. En revanche, les autres questions sont plus décousues.

Les réponses n'en sont pas moins pertinentes. La gestion de la mémoire dans les conteneurs peut être surprenante et les exceptions sont de retour. Le fait qu'il y ait moins de suite dans les questions les rend plus applicatives et moins génériques. Enfin, chaque question peut être utilisée telle quelle quand les questions en bloc nous entraînent dans la réflexion de plus en plus loin. C'est donc plus théorique que pratique ; ce n'est pas moins intéressant, c'est différent. Un très bon livre tout de même.

## IV - Liens annexes

 ***Critique sur la page de livres Jeux***

 ***Achat sur Amazon.fr***

 ***Lien vers le site de l'éditeur***



