

Exceptional C++

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

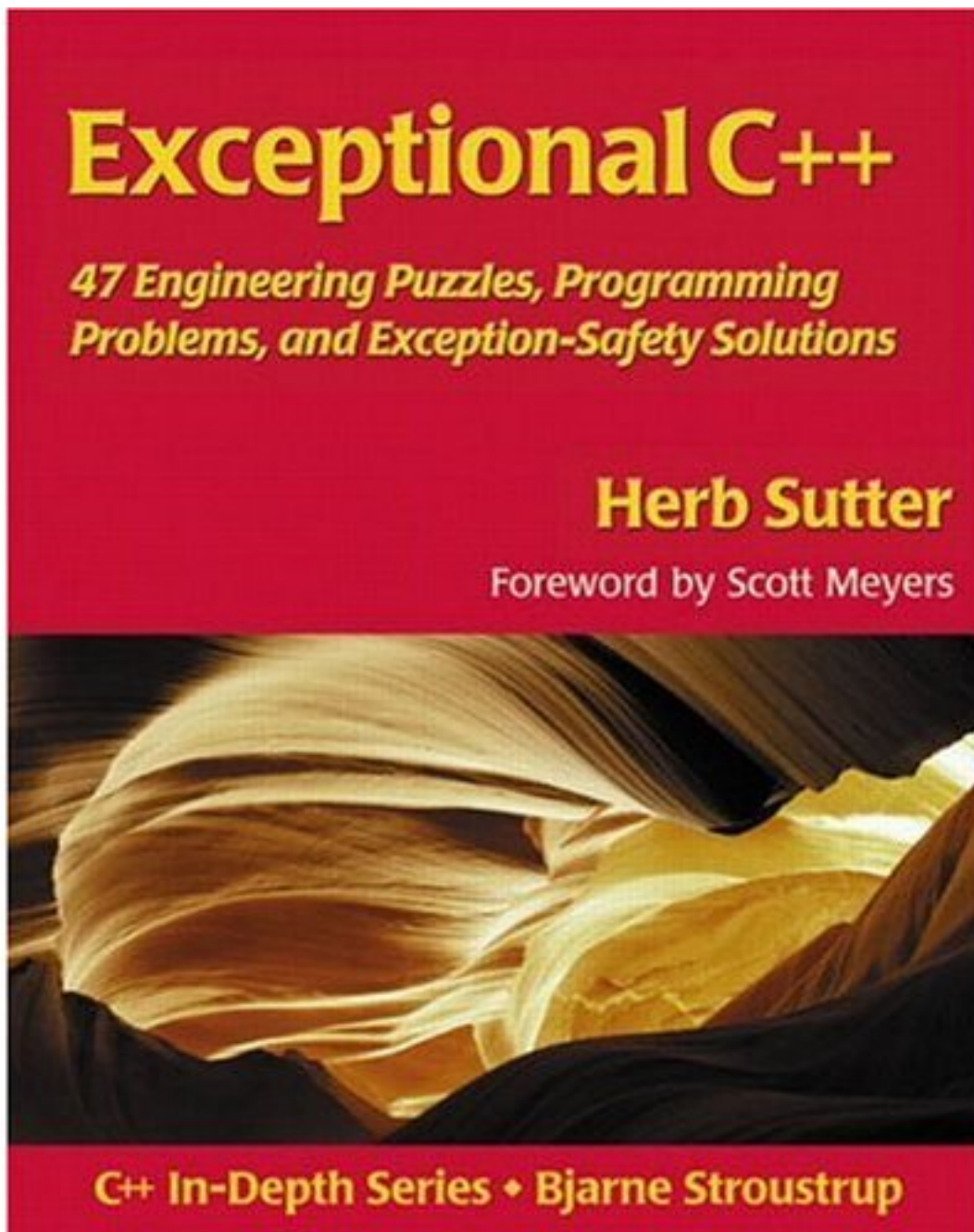
Date de publication : 08/05/2006

Dernière mise à jour : 27/03/2008

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

- I - Description
- II - Table des matières
- III - Critique : Indispensable
- IV - Liens annexes

I - Description



Une approche ludique à certains problèmes rencontrés en C++, tel est le pari de *Herb Sutter*. A partir de 47 questions posées sur Internet dans les séries *Guru Of The Week* auxquelles il invite à répondre soi-même dans un premier temps, il développe ses réponses, pertinentes et précises. Les arcanes du C++ sont abordés afin de pouvoir écrire du code robuste et efficace. Le niveau de difficulté de chaque question est précisé au début de chapitre afin de pouvoir se préparer et à la réflexion et à la réponse.

Le contenu du livre aborde tout d'abord les templates et la STL. Quels sont les risques à créer une classe de chaînes de caractères non sensibles à la casse ? Comment profiter au maximum de la STL ?

Le deuxième bloc de questions concerne les exceptions. Quand peut-on lancer une exception, quelles sont les exceptions que l'on peut lancer à quel moment, utiliser du code qui ne lancera jamais d'exception, ... Sans doute la partie du C++ la moins mise en valeur et la moins travaillée par chacun, les exceptions ne fonctionnent pas toujours comme on l'espère, et ces différences sont exposées dans ce livre.

La troisième partie concerne l'héritage et l'écriture élégante d'un design. Comment doit-on hériter et dans quelles proportions ? Devrait-on hériter plus souvent on peut-on s'en passer plus souvent qu'on ne le pense ?

Avec l'avènement de la STL, les temps de compilation ont augmenté. Est-il possible de les diminuer ? Comment peut-on le faire avec des idiomes nouveaux ?

Une partie est consacrée au principe de Koenig, le *Koenig lookup*, qui permet de résoudre des recherches de fonctions, mais ce principe n'est pas très connu, il mérite à le devenir plus.

Les trois dernières parties sont consacrées à la vie des variables et donc aussi à la gestion de la mémoire. Petit accent sur les nouvelles possibilités du C++, les `std::auto_ptr`, les initialisations de variables locales aux classes, les durées de vie, les conversions, ... Est aussi abordé le mot-clé `const` et son utilité, les casts, le type `bool`, sa vie, son oeuvre, ...

II - Table des matières

- **GENERIC PROGRAMMING AND THE C++ STANDARD LIBRARY.**
 - Item 1: Iterators.
 - Item 2: Case-Insensitive Strings-Part 1.
 - Item 3: Case-Insensitive Strings-Part 2.
 - Item 4: Maximally Reusable Generic Containers-Part 1.
 - Item 5: Maximally Reusable Generic Containers-Part 2.
 - Item 6: Temporary Objects.
 - Item 7: Using the Standard Library (or, Temporaries Revisited).
- **EXCEPTION-SAFETY ISSUES AND TECHNIQUES.**
 - Item 8: Writing Exception-Safe Code-Part 1.
 - Item 9: Writing Exception-Safe Code-Part 2.
 - Item 10: Writing Exception-Safe Code-Part 3.
 - Item 11: Writing Exception-Safe Code-Part 4.
 - Item 12: Writing Exception-Safe Code-Part 5.
 - Item 13: Writing Exception-Safe Code-Part 6.
 - Item 14: Writing Exception-Safe Code-Part 7.
 - Item 15: Writing Exception-Safe Code-Part 8.
 - Item 16: Writing Exception-Safe Code-Part 9.
 - Item 17: Writing Exception-Safe Code-Part 10.
 - Item 18: Code Complexity-Part 1.
 - Item 19: Code Complexity-Part 2.
- **CLASS DESIGN AND INHERITANCE.**
 - Item 20: Class Mechanics.
 - Item 21: Overriding Virtual Functions.
 - Item 22: Class Relationships-Part 1.
 - Item 23: Class Relationships-Part 2.
 - Item 24: Uses and Abuses of Inheritance.
 - Item 25: Object-Oriented Programming.
- **COMPILER FIREWALLS AND THE PIMPL IDIOM.**
 - Item 26: Minimizing Compile-time Dependencies-Part 1.
 - Item 27: Minimizing Compile-time Dependencies-Part 2.
 - Item 28: Minimizing Compile-time Dependencies-Part 3.
 - Item 29: Compilation Firewalls.
 - Item 30: The "Fast Pimpl" Idiom.
- **NAME LOOKUP, NAMESPACES, AND THE INTERFACE PRINCIPLE.**
 - Item 31: Name Lookup and the Interface Principle-Part 1.
 - Item 32: Name Lookup and the Interface Principle-Part 2.
 - Item 33: Name Lookup and the Interface Principle-Part 3.
 - Item 34: Name Lookup and the Interface Principle-Part 4.
- **MEMORY MANAGEMENT.**

- Item 35: Memory Management-Part 1.
- Item 36: Memory Management-Part 2.
- Item 37: auto_ptr.
- TRAPS, PITFALLS, AND ANIT-IDIOMS.
 - Item 38: Object Identity.
 - Item 39: Automatic Conversions.
 - Item 40: Object Lifetimes-Part 1.
 - Item 41: Object Lifetimes-Part 2.
- MISCELLANEOUS TOPICS.
 - Item 42: Variable Initialization#Or Is It?
 - Item 43: Const-Correctness.
 - Item 44: Casts.
 - Item 45: bool.
 - Item 46: Forwarding Functions.
 - Item 47: Control Flow.

III - Critique : Indispensable

La note maximale est largement méritée pour ce livre très agréable à lire et pour les explications claires. Certaines parties permettent de répondre à des questions qu'on a pu se poser, par exemple les chapitres sur les *namespaces* pour la résolution des fonctions à utiliser. D'autres abordent le problème des fuites mémoire. Les templates sont abordés pragmatiquement et on a une idée immédiate de leur puissance mais aussi des problèmes qu'ils peuvent apporter. Un accent est mis sur l'inutilité de l'optimisation aveugle, c'est un point à souligner et qui montre une fois de plus que *Sutter* maîtrise parfaitement son sujet.

Le livre a été modifié par rapport aux questions/réponses sur le net afin d'être conforme au standard C++. 2 suites sont aussi sorties. Sans doute LE livre pour arrêter de programmer en C avec du C++.

IV - Liens annexes

 ***Critique sur la page de livres C++***

 ***Achat sur Amazon.fr***

 ***Lien vers le site de l'éditeur***

