

Exceptional C++ Style

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

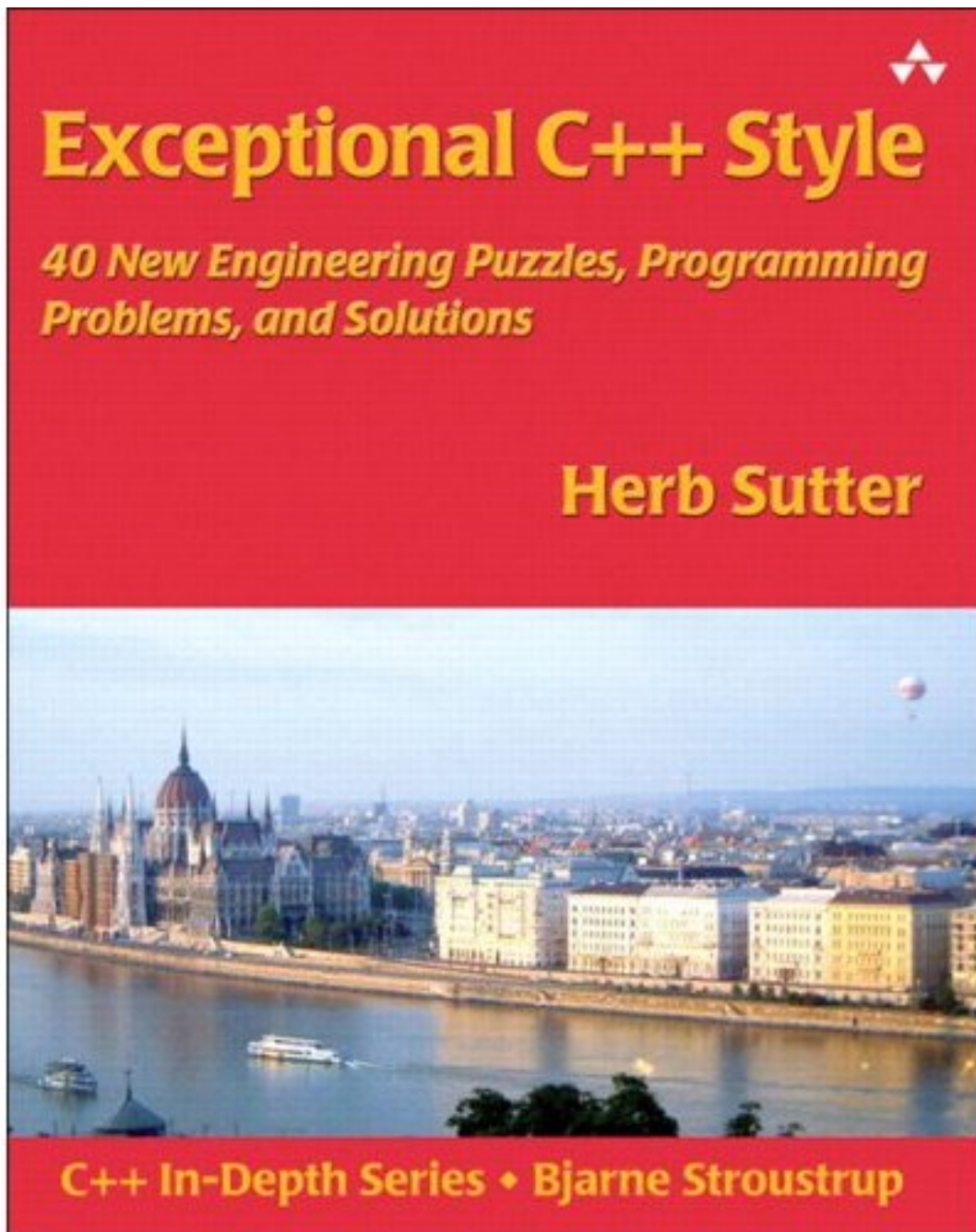
Date de publication : 08/06/2006

Dernière mise à jour : 27/03/2008

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

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

I - Description



Le "style" d'un logiciel consiste à trouver l'équilibre entre ce qu'on appelle l' *overhead* et la fonctionnalité, l'élégance et la maintenabilité, la flexibilité et l'excès. Dans Exceptional C++ Style , le gourou légendaire du C++, *Herb Sutter* présente 40 nouveaux scénarios de programmation afin d'analyser non seulement le quoi, mais aussi le pourquoi et de vous aider à trouver l'équilibre dans votre logiciel.

Organisé autour de problèmes et solutions pratiques, ce livre offre une nouvelle vision de détails cruciaux du C++ et de leur relations et de nouvelles stratégies pour les techniques de programmation clé du C++ actuel - y compris la programmation générique, la STL, les exceptions. Vous trouverez des réponses à des questions telles que :

- Que peut-on apprendre à propos du design de bibliothèques à partir de la STL ?
- Comment éviter de créer du code templaté non générique ?
- Pourquoi ne pas spécialiser des fonctions templatées ? Que faire à la place ?
- Comment la gestion des exceptions peut-elle aller au-delà des instructions try et catch ?
- Quand et comment exposer les parties privées d'une classe ?
- Comment créer des classes plus sûres pour le contrôle des versions ?
- Quel est le coût réel de l'utilisation des conteneurs standards ?
- Comment l'utilisation de const optimise réellement votre programme ?
- Comment le mot-clé inline affecte-t-il réellement la performance ?
- Quand le code qui semble faux compile et s'exécute parfaitement et pourquoi s'en soucier ?
- Qu'est qui est mauvais dans le design des `std::string` ?

Exceptional C++ Style vous aidera à créer, mettre en place l'architecture et coder avec style tout en permettant plus de robustesse et de performance dans tous vos logiciels C++.

(Traduction du dessous de couverture)

II - Table des matières

- **GENERIC PROGRAMMING AND THE C++ STANDARD LIBRARY.**
 - 1. Uses and Abuses of vector.
 - 2. The String Formatters of Manor Farm, Part 1: sprintf.
 - 3. The String Formatters of Manor Farm, Part 2: Standard (or Blindingly Elegant) Alternatives.
 - 4. Standard Library Member Functions.
 - 5. Flavors of Genericity, Part 1: Covering the Basis [sic].
 - 6. Flavors of Genericity, Part 2: Generic Enough?
 - 7. Why Not Specialize Function Templates?
 - 8. Befriending Templates.
 - 9. Export Restrictions, Part 1: Fundamentals.
 - 10. Export Restrictions, Part 2: Interactions, Usability Issues, and Guidelines.
- **EXCEPTION SAFETY ISSUES AND TECHNIQUES.**
 - 11. Try and Catch Me.
 - 12. Exception Safety: Is It Worth It?
 - 13. A Pragmatic Look at Exception Specifications.
- **CLASS DESIGN, INHERITANCE, AND POLYMORPHISM.**
 - 14. Order, Order!
 - 15. Uses and Abuses of Access Rights.
 - 16. (Mostly) Private.
 - 17. Encapsulation.
 - 18. Virtuality.
 - 19. Enforcing Rules for Derived Classes.
- **MEMORY AND RESOURCE MANAGEMENT.**
 - 20. Containers in Memory, Part 1: Levels of Memory Management.
 - 21. Containers in Memory, Part 2: How Big Is It Really?
 - 22. To new, Perchance to throw, Part 1: The Many Faces of new.
 - 23. To new, Perchance to throw, Part 2: Pragmatic Issues in Memory Management.
- **OPTIMIZATION AND EFFICIENCY.**
 - 24. Constant Optimization?
 - 25. inline Redux.
 - 26. Data Formats and Efficiency, Part 1: When Compression Is the Name of the Game.
 - 27. Data Formats and Efficiency, Part 2: (Even Less) Bit-Twiddling.
- **TRAPS, PITFALLS, AND PUZZLERS.**
 - 28. Keywords That Aren't (or, Comments by Another Name).
 - 29. Is It Initialization?
 - 30. double or Nothing.
 - 31. Amok Code.
 - 32. Slight Typos? Graphic Language and Other Curiosities.
 - 33. Operators, Operators Everywhere.
- **STYLE CASE STUDIES.**

- 34. Index Tables.
- 35. Generic Callbacks.
- 36. Construction Unions.
- 37. Monoliths "Unstrung," Part 1: A Look at std::string.
- 38. Monoliths "Unstrung," Part 2: Refactoring std::string.
- 39. Monoliths "Unstrung," Part 3: std::string Diminishing.
- 40. Monoliths "Unstrung," Part 4: std::string Redux.

III - Critique : Très utile

Troisième opus, et ça se sent. Des piqûres de rappel des 2 précédents livres, mais on sent un essoufflement certain. Les questions sont encore plus décousues - sauf la fin sur les `std::string` - et les réponses sont parfois paradoxales par rapport à ce qui était dit par le passé.

Par exemple, l'histoire des *const* dans ce livre souligne principalement qu'il ne sert pas au compilateur pour optimiser le code, mais plutôt au programmeur à faire des optimisations lui-même lorsqu'il sait que le code le permet, même s'il indique que les futurs compilateurs pourraient le permettre. D'ailleurs, il parle pas mal du futur, édition des liens globale, compilation à la volée, ... même de tentative infructueuses de recréer des unions dans le C++ mais que la tentative a échoué de peu car les compilateurs ne sont pas assez puissants. On sent peut-être une certaine justification ou correction par rapport aux livres précédents où les gens ont mal compris certaines solutions, ont conclu trop vite par rapport à des avantages, atouts des solutions données, alors qu'il ne voulait clairement pas aller au-delà de ce qu'il disait, les gens ont tenté de lire entre les lignes quelque chose qui n'existait pas. Ce livre permet de recadrer, à mon avis, l'excitation qui a pu surgir des 2 premiers livres.

IV - Liens annexes

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

 ***Achat sur Amazon.fr***

 ***Lien vers le site de l'éditeur***

