Checks based on the book "Effective C++ Third Edition: 55 Specific Ways to Improve Your Programs and Designs" by Scott Meyers

Amazon.com Purchase link: http://amzn.com/0321334876

> "Every C++ professional needs a copy of *Effective C++*. It is an absolute must-read for anyone thinking of doing serious C++ development. If you've never read *Effective C++* and you think you know everything about C++, think again."
> — **Steve Schirripa,** Software Engineer, Google

> "C++ and the C++ community have grown up in the last fifteen years, and the third edition of *Effective C++* reflects this. The clear and precise style of the book is evidence of Scott's deep insight and distinctive ability to impart knowledge."
> — **Gerhard Kreuzer,** Research and Development Engineer, Siemens AG

The first two editions of *Effective C++* were embraced by hundreds of thousands of programmers worldwide. The reason is clear: Scott Meyers' practical approach to C++ describes the rules of thumb used by the experts — the things they almost always do or almost always avoid doing — to produce clear, correct, efficient code.

The book is organized around 55 specific guidelines, each of which describes a way to write better C++. Each is backed by concrete examples. For this third edition, more than half the content is new, including added chapters on managing resources and using templates. Topics from the second edition have been extensively revised to reflect modern design considerations, including exceptions, design patterns, and multithreading.

Important features of *Effective C++* include:

- Expert guidance on the design of effective classes, functions, templates, and inheritance hierarchies.
- Applications of new "TR1" standard library functionality, along with comparisons to existing standard library components.
- Insights into differences between C++ and other languages (e.g., Java, C#, C) that help developers from those languages assimilate "the C++ way" of doing things.

From the Back Cover

"Every C++ professional needs a copy of *Effective C++*. It is an absolute must-read for anyone thinking of doing serious C++ development. If you've never read *Effective C++* and you think you know everything about C++, think again."
— **Steve Schirripa,** Software Engineer, Google"C++ and the C++ community have grown up in the last fifteen years, and the third edition of *Effective C++* reflects this. The clear and precise style of the book is evidence of Scott's deep insight and distinctive ability to impart knowledge."
— **Gerhard Kreuzer,** Research and Development Engineer, Siemens AG

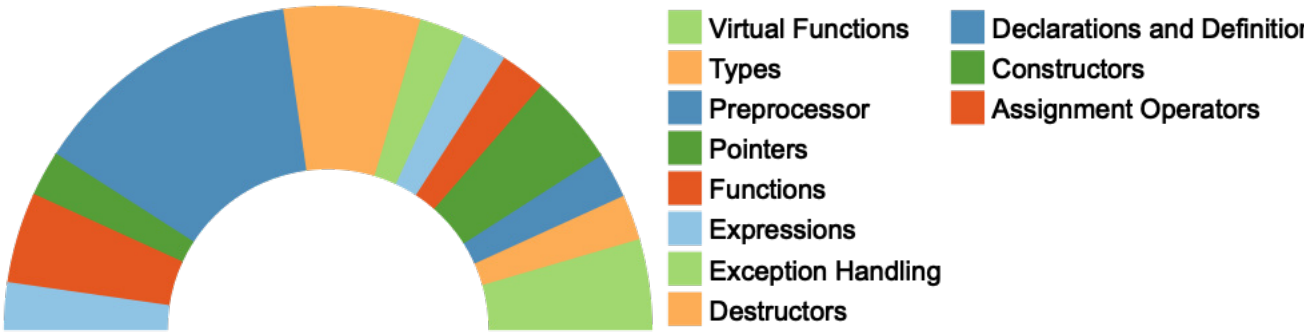
The first two editions of *Effective C++* were embraced by hundreds of thousands of programmers worldwide. The reason is clear: Scott Meyers' practical approach to C++ describes the rules of thumb used by the experts — the things they almost always do or almost always avoid doing — to produce clear, correct, efficient code.

The book is organized around 55 specific guidelines, each of which describes a way to write better C++. Each is backed by concrete examples. For this third edition, more than half the content is new, including added chapters on managing resources and using templates. Topics from the second edition have been extensively revised to reflect modern design considerations, including exceptions, design patterns, and multithreading.

Important features of *Effective C++* include:

- Expert guidance on the design of effective classes, functions, templates, and inheritance hierarchies.
- Applications of new "TR1" standard library functionality, along with comparisons to existing standard library components.
- Insights into differences between C++ and other languages (e.g., Java, C#, C) that help developers from those languages assimilate "the C++ way" of doing things.

Checks by Tags



Checks

Check ID	Check Name	Supported
EFFECTIVECPP_1	1. View C++ as a federation of languages	No
EFFECTIVECPP_02	2. Do Not Use #define	Yes
EFFECTIVECPP_03	3. Use Const whenever possible	Yes
EFFECTIVECPP_04	4. Make sure that objects are initialized before they are used	Yes
EFFECTIVECPP_5	5. Know what functions C++ silently writes and calls	No
EFFECTIVECPP_6	6. Explicitly disallow the use of compiler-generated functions you do not want	No
EFFECTIVECPP_07	7. Non-Virtual Destructors in Base Classes	Yes
EFFECTIVECPP_08	8. Exceptions in Destructors	Yes
EFFECTIVECPP_09	9. Virtual Call in Constructor/Destructor	Yes
EFFECTIVECPP_10	10. Assignment Operator Return This	Yes
EFFECTIVECPP_11	11. Assignment Operator Self Assignment	Yes
EFFECTIVECPP_12	12. Copy all parts of an object	No
EFFECTIVECPP_13	13. Use objects to manage resources	No
EFFECTIVECPP_14	14. Think carefully about copying behavior in resource-managing classes	No
EFFECTIVECPP_15	15. Provide access to raw resources in resource-managing classes	No
EFFECTIVECPP_16	16. Use the same form in corresponding uses of new and delete	Yes
EFFECTIVECPP_17	17. Store newed objects in smart pointers in standalone statements	Yes
EFFECTIVECPP_18	18. Make interfaces easy to use correctly and hard to use incorrectly	No
EFFECTIVECPP_19	19. Treat class design as type design	No
EFFECTIVECPP_20	20. Prefer pass-by-reference-to-const to pass by value	Yes
EFFECTIVECPP_21	21. Dont try to return a reference when you must return an object	No
EFFECTIVECPP_22	22. Datamembers should be declared private	Yes
EFFECTIVECPP_23	23. Prefer non-member non-friend functions to member functions	No
EFFECTIVECPP_24	24. Declare non-member functions when type conversions should apply to all parameters	No
EFFECTIVECPP_25	25. Consider support for a non-throwing swap	No

EFFECTIVECPP_26	26. Postpone variable definitions as long as possible	Yes
EFFECTIVECPP_27	27. Minimize casting	Yes
EFFECTIVECPP_28	28. Avoid returning “handles” to object internals	No
EFFECTIVECPP_29	29. Strive for exception-safe code	No
EFFECTIVECPP_30	30. Understand the ins and outs of inlining	No
EFFECTIVECPP_31	31. Minimize compilation dependencies between files	No
EFFECTIVECPP_32	32. Make sure public inheritance models "is-a"	No
EFFECTIVECPP_33	33. Avoid hiding inherited names	Yes
EFFECTIVECPP_34	34. Differentiate between inheritance of interface and inheritance of implementation	No
EFFECTIVECPP_35	35. Consider alternatives to virtual functions	Yes
EFFECTIVECPP_36	36. Never redefine an inherited non-virtual function	Yes
EFFECTIVECPP_37	37. Never redefine a (virtual) functions inherited default parameter value	No
EFFECTIVECPP_38	38. Model “has-a” or “is-implemented-in-terms-of” through composition	No
EFFECTIVECPP_39	39. Use private inheritance judiciously	No
EFFECTIVECPP_40	40. Use multiple inheritance judiciously	No
EFFECTIVECPP_41	41. Understand implicit interfaces and compile-time polymorphism	No
EFFECTIVECPP_42	42. Understand the two meanings of typename	No
EFFECTIVECPP_43	43. Know how to access names in templatized base classes	No
EFFECTIVECPP_44	44. Factor parameter-independent code out of templates	No
EFFECTIVECPP_45	45. Use member function templates to accept "all compatible types"	No
EFFECTIVECPP_46	46. Define non-member functions inside templates when type conversions are desired	No
EFFECTIVECPP_47	47. Use traits classes for information about types	No
EFFECTIVECPP_48	48. Be aware of template metaprogramming	No
EFFECTIVECPP_49	49. Understand the behavior of the new-handler	No
EFFECTIVECPP_50	50. Understand when it makes sense to replace new and delete	No
EFFECTIVECPP_51	51. Adhere to convention when writing new and	No

	delete	
EFFECTIVECPP_52	52. Write placement delete if you write placement new	No
EFFECTIVECPP_53	53. Pay attention to compiler warnings	No
EFFECTIVECPP_54	54. Familiarize yourself with the standard library, including TR1	No
EFFECTIVECPP_55	55. Familiarize yourself with Boost	No