

BARTŁOMIEJ FILIPEK

# C++17 IN DETAIL

LEARN WHAT ARE THE EXCITING FEATURES OF  
THE NEW STANDARD!

[BFILIPEK.COM](http://BFILIPEK.COM)

# C++17 in Detail

Learn what are the Exciting Features of The New C++ Standard!

Bartłomiej Filipek

This book is for sale at <http://leanpub.com/cpp17indetail>

This version was published on 2018-08-05



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2018 Bartłomiej Filipek

# Contents

<b>About the Author</b> . . . . .	<b>i</b>
<b>Revision History</b> . . . . .	<b>ii</b>
<b>Preface</b> . . . . .	<b>iii</b>
<b>About the Book</b> . . . . .	<b>iv</b>
Who This Book is For . . . . .	iv
Overall Structure of the Book . . . . .	v
Reader Feedback . . . . .	vi
Example Code . . . . .	vi
<b>Part 1 - The Language Features</b> . . . . .	<b>1</b>
Quick Start . . . . .	2
<b>1. Fixes and Deprecation</b> . . . . .	<b>4</b>
Removed Things . . . . .	5
Fixes . . . . .	9
Compiler support . . . . .	11
<b>2. Language Clarification</b> . . . . .	<b>12</b>
Stricter expression evaluation order . . . . .	13
Guaranteed Copy Elision . . . . .	17
Dynamic Memory Allocation for Over-Aligned Data . . . . .	22
Exception Specifications Part of the Type System . . . . .	23
Compiler Support . . . . .	23
<b>3. General Language Features</b> . . . . .	<b>24</b>
Structured Binding Declarations . . . . .	25
Init Statement for if/switch . . . . .	30
Inline Variables . . . . .	32
constexpr Lambda Expressions . . . . .	34
Compiler support . . . . .	35
<b>4. Templates</b> . . . . .	<b>36</b>

Template Argument Deduction for Class Templates . . . . .	37
Fold Expressions . . . . .	40
if constexpr . . . . .	43
Declaring Non-Type Template Parameters With auto . . . . .	50
Other Changes . . . . .	51
Compiler Support . . . . .	51
5. <b>Standard Attributes</b> . . . . .	<b>52</b>
Why Do We Need Attributes? . . . . .	53
Before C++11 . . . . .	53
Attributes in C++11 and C++14 . . . . .	54
C++17 additions . . . . .	56
Section Summary . . . . .	61
Compiler support . . . . .	61
<b>Part 2 - The Standard Library Changes</b> . . . . .	<b>63</b>
6. <b>std::optional</b> . . . . .	<b>64</b>
Introduction . . . . .	65
std::optional Creation . . . . .	67
Returning std::optional . . . . .	71
Accessing The Stored Value . . . . .	72
std::optional Operations . . . . .	73
Examples of std::optional . . . . .	76
Performance & Memory Consideration . . . . .	78
Migration from boost::optional . . . . .	79
Special case: optional<bool> and optional<T*> . . . . .	80
Summary . . . . .	81
Compiler Support . . . . .	81
7. <b>std::variant</b> . . . . .	<b>82</b>
The Basics . . . . .	83
std::variant Creation . . . . .	86
Changing the Values . . . . .	89
Accessing the Stored Value . . . . .	91
Visitors for std::variant . . . . .	92
Other std::variant Operations . . . . .	94
Exception Safety Guarantees . . . . .	95
Performance & Memory Considerations . . . . .	96
Migration From boost::variant . . . . .	97
Examples of std::variant . . . . .	97
Wrap Up . . . . .	104
Compiler Support . . . . .	104

## CONTENTS

<b>8. <code>std::any</code></b>	<b>105</b>
The Basics	106
<code>std::any</code> Creation	108
Changing the Value	110
Accessing The Stored Value	111
Performance & Memory Considerations	112
Migration from <code>boost::any</code>	113
Examples of <code>std::any</code>	113
Wrap Up	116
Compiler Support	116
<b>9. <code>std::string_view</code></b>	<b>117</b>
The Basics	118
The <code>std::basic_string_view</code> Type	119
<code>std::string_view</code> Creation	120
Other Operations	121
Risks With Using <code>string_view</code>	123
Performance & Memory Considerations	132
Migration from <code>boost::string_ref</code> and <code>boost::string_view</code>	133
Wrap Up	134
<b>10. String Operations</b>	<b>135</b>
<b>11. Filesystem</b>	<b>136</b>
Filesystem Overview	137
Examples	138
Section Summary	142
Compiler Support	142
<b>12. Parallel Algorithms</b>	<b>143</b>
Introduction	144
Overview	145
Execution policies	145
Algorithm update	147
New algorithms	149
Summary	150
Compiler Support	151
<b>13. Other Changes In The Library</b>	<b>152</b>
<code>std::byte</code>	153
Improvements for Maps and Sets	154
Return Type of Emplace Methods	157
Sampling Algorithms	158
Special Mathematical Functions	159

## CONTENTS

Shared Pointers and Arrays	160
Non-member <code>size()</code> , <code>data()</code> and <code>empty()</code>	161
<code>constexpr</code> Additions to the Standard Library	161
Compiler support	163
<b>Part 3 - More Examples and Use Cases</b>	<b>164</b>
<b>14. Refactoring with <code>std::optional</code></b>	<b>165</b>
The Use Case	166
The Tuple Version	167
A Separate Structure	168
With <code>std::optional</code>	169
With <code>std::variant</code>	170
Wrap up	172
<b>15. Enforcing Code Contracts With <code>[[nodiscard]]</code></b>	<b>173</b>
Introduction	174
Where Can It Be Used?	174
How to Ignore <code>[[nodiscard]]</code>	177
Before C++17	177
Summary	178
<b>16. Replacing <code>enable_if</code> with <code>if constexpr</code> - Factory with Variable Arguments</b>	<b>179</b>
The Problem	180
Before C++17	182
With <code>if constexpr</code>	183
Summary	184
<b>Appendix A - Compiler Support</b>	<b>185</b>
GCC	185
Clang	185
VisualStudio - MSVC	185
Compiler Support of C++17 Features	186
<b>Appendix B - Resources and References</b>	<b>188</b>