

Table of Contents

Notes by Standard at a Glance	19
Notes belonging to C++11	19
Notes belonging to C++17	20
Notes belonging to C++20	21
1 Tips and Tricks with Templates	23
Note 1: Know the name	25
Note 2: Templates can have type and non-type parameters . . .	27
Note 3: When to use <code>typename</code> , when <code>class</code>	31
Note 4: The parts of a variadic template	33
Note 5: There are no implicit conversions for template parameters	37
Note 6: Alias template for clean Template Meta-Programming (TMP)	41
Note 7: Variable template for clean TMP	43
Note 8: The trailing-return-type with <code>decltype</code> and <code>void()</code>	45
Note 9: Use <code>decltype</code> when you need to construct a type for test- ing during compile-time	49
Note 10: What <code>void_t</code> does	51
Note 11: Keep that array's size	53
Note 12: There is no else if in C++	57
Note 13: More useful than it appears: <code>always_false</code>	61
Note 14: Prefer <code>auto</code> as non-type template parameter (NTTP) to reduce redundancy	63
Note 15: Block template argument deduction	65

Note 16: Fold expressions and the comma operator	69
Note 17: Poor men's fold expressions	73
Note 18: From an array to a pack	75
Note 19: From an array to a pack with templated lambda	77
Note 20: Create objects in place for direct use	79
Note 21: Guidelines for efficient use of templates	85
Note 22: Put <code>enable_if</code> on the return type	91
Note 23: <code>enable_if</code> and how to disable a member function	93
Note 24: How to disable a special member function	95
Acronyms	99
Bibliography	101
Index	103

Notes belonging to C++11

Note 4: The parts of a variadic template	33
Note 6: Alias template for clean TMP	41
Note 8: The trailing-return-type with decltype and void()	45
Note 9: Use declval when you need to construct a type for testing during compile-time	49
Note 10: What void_t does	51
Note 13: More useful than it appears: always_false	61
Note 15: Block template argument deduction	65
Note 17: Poor men's fold expressions	73
Note 22: Put enable_if on the return type	91
Note 23: enable_if and how to disable a member function	93
Note 24: How to disable a special member function	95

Notes belonging to C++17

Note 12: There is no else if in C++	57
Note 14: Prefer auto as NTTP to reduce redundancy	63
Note 16: Fold expressions and the comma operator	69
Note 18: From an array to a pack	75
Note 20: Create objects in place for direct use	79

Notes belonging to C++20

Note 19: From an array to a pack with templated lambda 77