

# C++ Club UK

Gleb Dolgich

2019-08-22

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/#mailing2019-08>

► [Reddit](#)

Arthur O'Dwyer

<https://quuxplusone.github.io/blog/2019/08/18/post-cologne-mailing/>

► [Reddit](#)

**Reddit:** Is the C++ committee or any key influencers in the C++ community working on anything to tackle the advantages that Rust has over C++ (eg. Rust's borrow checking, compiler-checked concurrency safety and cargo package management)

<https://devblogs.microsoft.com/cppblog/vcpkg-2019-07-update/>

[https://www.reddit.com/r/cpp/comments/cqta79/vcpkg\\_201907\\_update/](https://www.reddit.com/r/cpp/comments/cqta79/vcpkg_201907_update/)

*Hopefully you will not require specific versions of packages, as the workflow of vcpkg doesn't really support the (very common) scenario well.*

# Boost 1.71 released

[https://www.boost.org/users/history/version\\_1\\_71\\_0.html](https://www.boost.org/users/history/version_1_71_0.html)

- ▶ NEW: **Variant2**: A never-valueless, strong guarantee implementation of `std::variant`, from Peter Dimov.
- ▶ [Reddit](#)

# Lambdas vs. Closures

Scott Meyers

► [Reddit](#)

# Web Framework Benchmarks

<https://www.techempower.com/benchmarks/#section=test&runid=26a79c95-5eec-4572-8c94-dd710df659d7&hw=ph&test=update>

► Reddit

<https://github.com/an-tao/drogon>

**Drogon:** A C++14/17 based HTTP web application framework running on Linux/macOS/Unix

## Barry Revzin

- ▶ What is unified function call syntax anyway?
  - ▶ Reddit
- ▶ UFCS Customization and Extension
  - ▶ Reddit
  - ▶ P1282R0 Ceci N'est Pas Une Pipe: Adding a workflow operator to C++



## Bjarne Stroustrup

### ► A bit of background for the unified call proposal

Based on real input from code and users, I reluctantly agreed that for compatibility reasons,  $x.f(y)$  and  $f(x,y)$  could not mean exactly the same. The only feasible way forward was to do a traditional lookup based on the syntax used, and then try the other syntax if the first one failed. Stability – backwards compatibility – is an important feature, overruling my desire for perfection.

### ► P0131 Unified call syntax concerns

# Introducing the Rule of DesDeMovA (1/4)

Blog post by Peter Sommerlad

<https://blog.safecpp.com/2019/07/01/initial.html>

[https://accu.org/content/conf2014/Howard\\_Hinnant\\_Accu\\_2014.pdf](https://accu.org/content/conf2014/Howard_Hinnant_Accu_2014.pdf)

Rule of Zero:

*Code that you do not write cannot be wrong.*

# Introducing the Rule of DesDeMovA (2/4)

C++ now

2019  
MAY 6-10  
cppnow.org



Peter Sommerlad

Rule of DesDeMovA

Video Sponsorship  
Provided By:



03:54

Voting Closed

Do you Remember: What Special Member Functions Do You Get?

3

What you write	What you get					
	default constructor	destructor	copy constructor	copy assignment	move constructor	move assignment
nothing	defaulted	defaulted	defaulted	defaulted	defaulted	defaulted
any constructor	not declared	defaulted	defaulted	defaulted	defaulted	defaulted
default constructor	user declared	defaulted	defaulted	defaulted	defaulted	defaulted
destructor	defaulted	user declared	defaulted (!)	defaulted (!)	not declared	not declared
copy constructor	not declared	defaulted	user declared	defaulted (!)	not declared	not declared
copy assignment	defaulted	defaulted	defaulted (!)	user declared	not declared	not declared
move constructor	not declared	defaulted	deleted	deleted	user declared	not declared
move assignment	defaulted	defaulted	deleted	deleted	not declared	user declared

Howard Hinnant's Table: [https://ericniebler.com/2014/05/14/Howard\\_Hinnant\\_Accu\\_2014.pdf](https://ericniebler.com/2014/05/14/Howard_Hinnant_Accu_2014.pdf)

Note: Declaring the defaulted special members denoted with a (!) is a bug in the standard.

© Peter Sommerlad

# Introducing the Rule of DesDeMovA (3/4)

C++ now

2019  
MAY 6-10  
cppnow.org



Peter Sommerlad

Rule of DesDeMovA

Video Sponsorship  
Provided By:



02:38

Voting Closed

Rule of DesDeMovA: T&& operator=(T&&) noexcept=delete;

6


What you write	DesDeMovA Rule of if Destructor defined Deleted Move Assignment					
	default constructor	copy constructor	copy assignment	move constructor	move assignment	
nothing	defaulted					
any constructor	not declared					
default constructor	user declared	defaulted	defaulted	defaulted	defaulted	
destructor	defaulted	user declared	defaulted (!)	defaulted (!)	not declared	not declared
copy constructor	not declared	defaulted	user declared	defaulted (!)	not declared	not declared
copy assignment	defaulted	defaulted	defaulted (!)	user declared	not declared	not declared
move constructor	not declared	defaulted	deleted	deleted	user declared	not declared
move assignment	defaulted	user declared	deleted	deleted	not declared	=delete

Howard Hinnant's Table: [https://ericniebler.com/2014/hinnant\\_hinnant\\_2014.pdf](https://ericniebler.com/2014/hinnant_hinnant_2014.pdf)  
Note: Defining the defaulted special members denoted with a (!) is a bug in the standard.

# Introducing the Rule of DesDeMovA (3/4)

C++ now


2019  
MAY 6-10  
cppnow.org



**Peter Sommerlad**


Rule of DesDeMovA

Video Sponsorship  
Provided By:




02:19

Voting Closed


Summary  7

1. Rule of Zero
2. Rule of DesDeMovA (no copy, no move for SBRM/RAII and OO-Base classes)
3. Rule of Unique Resource Managers (move-only, no copy)
4. Rule of Five for Resource Managers with Value Semantics, or other really special cases



**Cevelop**  
Your C++ deserves it

Download IDE at:  
[www.cevelop.com](http://www.cevelop.com)



Sponsors  
welcome!

Commercial  
licensing  
possible!

## strong\_typedef - Create distinct types for distinct purposes

Article by Anthony Williams

[https://www.justsoftwaresolutions.co.uk/cplusplus/strong\\_typedef.html](https://www.justsoftwaresolutions.co.uk/cplusplus/strong_typedef.html)

[https://github.com/anthonywilliams/strong\\_typedef](https://github.com/anthonywilliams/strong_typedef)

```
1 using transaction_id =  
2     jss::strong_typedef<struct transaction_tag, std::string>;  
3  
4 bool is_a_foo(transaction_id id)  
5 {  
6     auto &s = id.underlying_value();  
7     return s.find("foo") != s.end();  
8 }
```

<https://www.cycfi.com/2019/07/photon-micro-gui/>

► [Reddit](#)

# C++ libraries for data visualization

- ▶ VTK <https://vtk.org/>
- ▶ ROOT <https://root.cern.ch/>
- ▶ matplotlib-cpp <https://github.com/lava/matplotlib-cpp>
  - ▶ matplotlib (Python) <https://matplotlib.org/>
- ▶ QCustomPlot (QT, GPL/commercial) <https://www.qcustomplot.com/>



<http://cppcast.com/2019/07/robert-maynard/>

► [Reddit](#)

## CMake line by line - creating a header-only library

<http://dominikberner.ch/cmake-interface-lib/>

► [Reddit](#)

<https://github.com/bernedom/SI>

Professional CMake: A Practical Guide, 4th ed., CMake 3.15

<https://crascit.com/professional-cmake/> \$30

# OSes built using C++

## Reddit

- ▶ TempleOS
- ▶ Haiku
- ▶ Google Fuchsia
- ▶ IncludeOS
- ▶ DistortOS (RTOS)
- ▶ Symbian OS (Dead)
- ▶ SerenityOS

# Agner Vector Class Library V2

This is a C++17 class library for using the Single Instruction Multiple Data (SIMD) instructions in modern microprocessors.

<https://www.agner.org/optimize/blog/read.php?i=1013>

<https://github.com/vectorclass/version2> (Apache 2.0)

Manual [https://github.com/vectorclass/manual/blob/master/vcl\\_manual.pdf](https://github.com/vectorclass/manual/blob/master/vcl_manual.pdf)

# Approval tests (1/2)

Also known as **Golden Master Tests** or **Snapshot Testing** (locking down current behaviour)

- ▶ CppOnSea 2019 - Clare Macrae - Quickly testing legacy code  
<https://youtu.be/dtm8V3TIB6k>
  - ▶ Slides <https://slideshare.net/ClareMacrae>
  - ▶ CppCast with Clare Macrae <https://cppcast.com/clare-macrae/>
    - ▶ r/cpp [https://www.reddit.com/r/cpp/comments/ckzc11/cppcast\\_approval\\_tests\\_with\\_clare\\_macrae/](https://www.reddit.com/r/cpp/comments/ckzc11/cppcast_approval_tests_with_clare_macrae/)
- ▶ Code <https://github.com/approvals/ApprovalTests.cpp> (Apache 2.0)
- ▶ Approval Tests Library - Capturing Human Intelligence [available for Java, C#, VB.Net, PHP, Ruby, Node.JS and Python] <https://approvaltests.com/> by Llevelyn Falco
  - ▶ Supports Catch, Catch 2, Google Test, Okra
- ▶ Mutation tests: sabotage the code
  - ▶ Mutate++ [https://github.com/nlohmann/mutate\\_cpp](https://github.com/nlohmann/mutate_cpp)

### Books

- ▶ Modern C++ Programming with Test-Driven Development, by Jeff Langr [[Safari Books Online](#)]
- ▶ Your Code as a Crime Scene, by Adam Tornhill [[Safari Books Online](#)]
- ▶ Software Design X-Rays, by Adam Tornhill [[Safari Books Online](#)]

### Tools

- ▶ OpenCoverage <https://github.com/OpenCppCoverage>
- ▶ BullseyeCoverage <https://www.bullseye.com>

- ▶ Implementation <https://github.com/kokkos/mdspan> (BSD 3-Clause)
  - ▶ Intro <https://github.com/kokkos/mdspan/wiki/A-Gentle-Introduction-to-mdspan>
  - ▶ r/cpp [https://www.reddit.com/r/cpp/comments/cl127i/mdspan\\_productionquality\\_reference\\_implementation/](https://www.reddit.com/r/cpp/comments/cl127i/mdspan_productionquality_reference_implementation/)
- ▶ Kokkos <https://github.com/kokkos/kokkos>
- ▶ Multi-dimensional strided array views in Magnum <https://blog.magnum.graphics/backstage/multidimensional-strided-array-views/>
- ▶ P0009R9 **mdspan**: A Non-Owning Multidimensional Array Reference <http://wg21.link/p0009r9>
- ▶ CppCast with Bryce Adelstein Lelbach <https://cppcast.com/bryce-lelbach-mdspan/>

<https://github.com/aras-p/ClangBuildAnalyzer>





**Tony Van Eerd** @tvaneerd

```
try {  
    return vec.at(index);  
}  
catch (std::out_of_range const &) {  
    return -1;  
}
```

I just saw this in a code review and was going to claim it inefficient, but darn [@CompileExplore](#) proved me wrong again. (IIRAC\*) All compilers turn it into an if check on the bounds.

\*Read ASM

11h • 11/07/2019 • 22:56





**Simon Brand** @TartanLlama

Threw every C++ proposal title in history into a recurrent neural network and some sound surprisingly legit

```
Introducing the Proposal for an Resumable memory related initialization from string_user enuming_values
Integer C++ Function (Revision 9)
Fixing The Konates of Capture
feature_lock_view-expressions
GENDA, Tiny Proposed Regrogish to Standard Library
Transparentic atomic operations (rev. 3)
Editor's Report for Deprecate Vector Technical Specification and Allocation
Containers for Parflock Points
std::function without a requirement and optimization for the C++2 Defects Record
Propose-a the Generic Memory Modification
Equalized Interaction for the Model Proposal for P0107R2
Introduce types in computing and string_that in C++
Contract Design Aliation for be Business
Gegropage initialization finesy policies in point memory scheduler be_rodng_robot and enumerations
Reflection of noexcept in C++20
std:::haluter_tokence_related (revision 1)
Meeting Not C++
Parallellative Intermate() and unlumbing_cip
What madened: Propose-tight member of Painfreacted Default
Lookup Asiomation TS
```

2d • 16/07/2019 • 20:07 •

