

C++ Club

Gleb Dolgich

2019-08-01

Trip Report: C++ Standards Meeting in Cologne, July 2019

Botond Ballo | [Reddit](#)

Contracts were pulled from C++20 because the consensus for their current design has disappeared¹

Guy Davidson | [Reddit](#)

The big news though was the withdrawal of Contracts from the working draft. The consensus is that it simply isn't ready yet <...>

¹<https://botondballo.wordpress.com/2019/07/26/trip-report-c-standards-meeting-in-cologne-july-2019/#contracts>

Survey results: Your “top five” ISO C++ feature proposals

<https://herbsutter.com/2019/07/25/survey-results-your-top-five-iso-c-feature-proposals/>

Comment:

Indeed, the results are incredibly biased, in favor of:

- ▶ *Your own proposals.*
- ▶ *Your own interests.*
- ▶ *Proposals made recently or which have gone through part of the adoption pipeline.*

https://www.reddit.com/r/cpp/comments/ci0uz6/survey_results_your_top_five_iso_c_feature/

- ▶ ELI5

Simplify Your Code With Rocket Science: C++20's Spaceship Operator

<https://devblogs.microsoft.com/cppblog/simplify-your-code-with-rocket-science-c20s-spaceship-operator/>

https://www.reddit.com/r/cpp/comments/c68457/simplify_your_code_with_rocket_science_c20s/

How do you get the benefits of Rust in C++?

https://www.reddit.com/r/cpp/comments/c6gtd4/how_do_you_get_the_benefits_of_rust_in_c/

- ▶ SaferCPlusPlus <https://github.com/duneroadrunner/SaferCPlusPlus/blob/master/README.md>
- ▶ Clang 10 thread safety analysis
<https://clang.llvm.org/docs/ThreadSafetyAnalysis.html>
- ▶ Sanitizers <https://github.com/google/sanitizers>
- ▶ Escher C++ Verifier <http://www.eschertech.com/papers/ecvpp2016.pdf>
- ▶ Lifetime profile
 - ▶ Paper <https://github.com/isocpp/CppCoreGuidelines/blob/master/docs/Lifetime.pdf>
 - ▶ Clang implementation
<https://eurollvm2019.sched.com/event/MGhd/implementing-the-c-core-guidelines-lifetime-safety-profile-in-clang>
 - ▶ Lifetime Profile Update in Visual Studio 2019 Preview 2
<https://devblogs.microsoft.com/cppblog/lifetime-profile-update-in-visual-studio-2019-preview-2/>

How do C++ developers manage dependencies?

https://www.reddit.com/r/cpp/comments/c6l3eg/how_do_c_developers_manage_dependencies/

Through much pain and anguish.

Scott Meyers' TD trick

https://www.reddit.com/r/cpp/comments/c6vnb3/just_started_learning_c_coming_from_python_and/eshq8vb?utm_source=share&utm_medium=web2x

```
1 | template <typename T> struct TD; // no definition
```

Now you write something like `TD<decltype(thing)>` and the error message tells you the type of `thing` (as deduced by `decltype`, of course, but in this case that's probably what you want).

Just started learning C++ coming from Python

https://www.reddit.com/r/cpp/comments/c6vnb3/just_started_learning_c_coming_from_python_and/

The new GCC compiler with colour highlighting is a little bit better at pointing out errors. It's generally quite helpful for pure C/C++ until you make an error with the standard library and you get 200 lines about std:: whatever<random characters>

In C++ a trick I always use when the error message is massive is to just focus on the first error.

Use **constexpr** for faster, smaller, and safer code

<https://blog.trailofbits.com/2019/06/27/use-constexpr-for-faster-smaller-and-safer-code/>

https://www.reddit.com/r/cpp/comments/c646ng/use_constexpr_for_faster_smaller_and_safer_code/

<https://github.com/trailofbits/constexpr-everything> (Apache 2.0)

A closer look at **bake**: a tool that makes building C/C++ code effortless

<https://medium.com/@cortoproject/a-closer-look-at-bake-a-tool-that-makes-building-c-c-code-effortless-b2e0409fad8f>

- ▶ https://www.reddit.com/r/C_Programming/comments/a85f6w/meet_bake_a_new_build_system_package_manager_for/
- ▶ https://www.reddit.com/r/cpp/comments/a8d7ny/meet_bake_a_new_build_system_package_manager_for/
- ▶ <https://news.ycombinator.com/item?id=18787777>

<https://github.com/SanderMertens/bake> (GPLv3)

A cargo-like buildsystem and package manager for C/C++

Magic.

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

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/2016/06/06/Howard_Hinnant_August_2014.pdf

Note: Getting 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	default constructor		copy constructor		move constructor		move assignment	
	nothing	defaulted	nothing	defaulted	nothing	defaulted	nothing	defaulted
any constructor	not declared		not declared		not declared		not declared	
default constructor	user declared		user declared		user declared		user declared	
destructor	defaulted		user declared		user declared		user declared	
copy constructor	not declared		defaulted		defaulted		defaulted	
copy assignment	defaulted		defaulted		defaulted		defaulted	
move constructor	not declared		defaulted		defaulted		defaulted	
move assignment	defaulted		user declared		deleted		deleted	


DesDeMovA
Rule of if
Destructor defined
Deleted
Move Assignment

Howard Hinnant's Table: <https://ericniebler.com/2015/04/01/rule-of-five/>
Note: Getting the defaulted special members denoted with a (1) 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

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



Download IDE at:
www.cevloop.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://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/>

[https:](https://www.reddit.com/r/cpp/comments/ccq9pn/elemental_c_gui_library/)

[//www.reddit.com/r/cpp/comments/ccq9pn/elemental_c_gui_library/](https://www.reddit.com/r/cpp/comments/ccq9pn/elemental_c_gui_library/)

Are there any good 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/>

https://www.reddit.com/r/cpp/comments/c9bpxb/cppcast_cmake_and_vtk_with_robert_maynard/

CMake line by line - creating a header-only library

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

https://www.reddit.com/r/cpp/comments/c8ty2h/a_line_by_line_explanation_how_to_create_a/

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

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

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

Survey results: Your “top five” ISO C++ feature proposals

<https://herbsutter.com/2019/07/25/survey-results-your-top-five-iso-c-feature-proposals/>

"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable." - Leslie Lamport

