

C++ Club UK

Gleb Dolgich

2019-08-22

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/2018/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	Rule of 1		Rule of 2		Rule of 3	
	default constructor	copy constructor	move constructor	move assignment	copy assignment	destructor
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	user declared	deleted	deleted	not declared	=delete


DesDeMovA
Rule of if
Destructors defined
Deleted
Move Assignment

Howard Hinnant's Table: <https://ericniebler.com/2015/05/01/rule-of-five/>
Note: Getting 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



Download IDE at:
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://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

Are there any OSES built using C++

https://www.reddit.com/r/cpp/comments/cho1qb/are_there_any_oses_built_using_c/

- ▶ 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

Approval tests

Also known as **Golden Master Tests** or **Snapshot Testing**

- ▶ CppOnSea 2019 - Clare Macrae - Quickly testing legacy code
<https://youtu.be/dtm8V3TIB6k>
- ▶ CppCast with Clare Macrae <https://cppcast.com/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/>