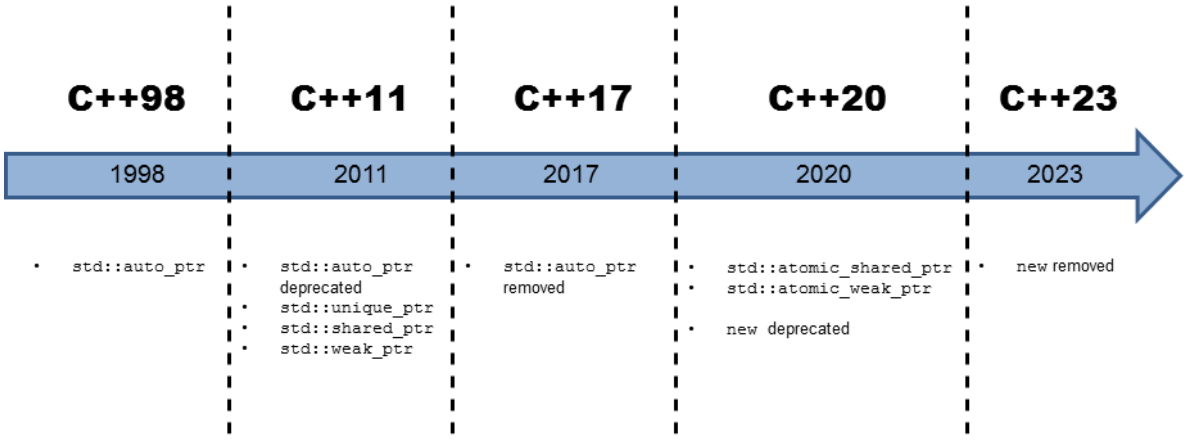
No New New: Raw Pointers Removed from C++

(http://www.facebook.com/rainer.grimm.
(https://plus.google.com/109576694736
(https://www.linkedin.com/in/rainergrimn
(https://twitter.com/rainer_grimm)
(https://www.xing.com/profile/Rainer_Gr

26 March 2018

Two weeks ago, the ISO C++ standard meeting took place in Jacksonville. Today I want to make a short detour and write about the revolutionary decision that was made in the Jacksonville meeting. Additionally, I refer to the post C++ Will no Longer Have Pointers (https://www.fluentcpp.com/2018/04/01/cpp-will-no-longer-have-pointers/) by Fluent C++. The standard committee decided that pointers will be deprecated in C++20 and will with very high probability be removed in C++23.

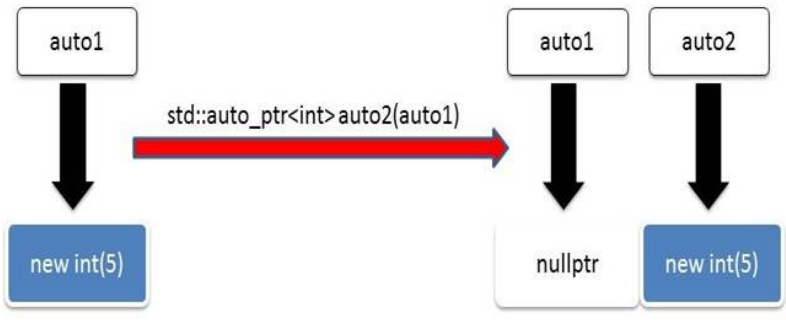
To be honest, what seems like a revolution is only the last step in a long evolution. First, let me explain the big picture.



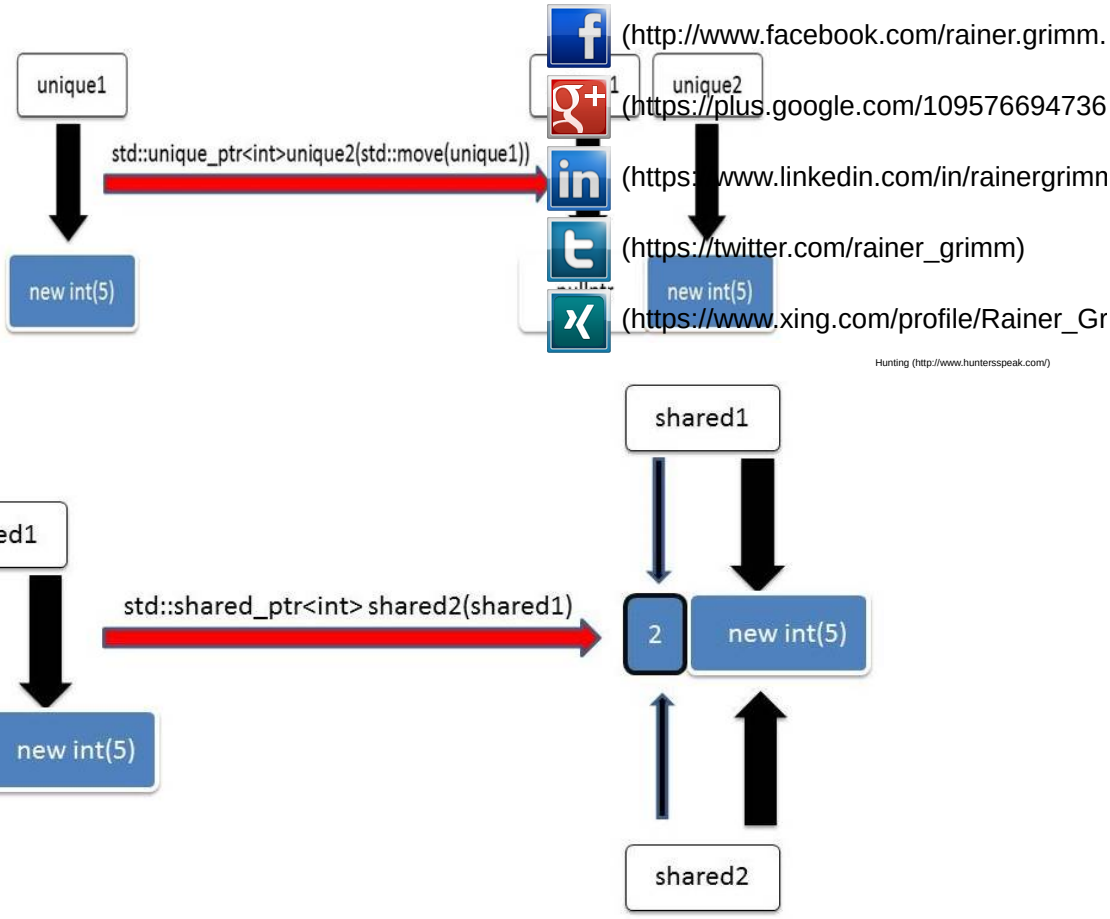
The evolution of pointers in C++

Pointers are part of C++ since the first beginning. We got them from C. From the first beginning, there was always the tendency in C++ to make the handling with pointers more type-safe without paying the extra cost.

With C++98 we got `std::auto_ptr` to express exclusive ownership. But `std::auto_ptr` had a big issue. When you copy a `std::auto_ptr` the resource will be moved. What looks like a copy operation was actually a move operation. The graphic shows the surprising behaviour of an `std::auto_ptr`.



This was extremely bad and the reason for a lot of serious bugs; therefore, we got `std::unique_ptr` with C++11 and `std::auto_ptr` was deprecated in C++11 and finally removed in C++17. Additionally, we got `std::shared_ptr` and `std::weak_ptr` in C++11 for handling shared ownership. You can not copy but move an `std::unique_ptr` and if you copy or assign an `std::shared_ptr`, the internal reference counter will be increased. Have a look here:



Since C++11 C++ has a multithreading library. This makes the handling with `std::shared_ptr` quite challenging because a `std::shared_ptr` is per definition shared but not thread-safe. Only the control-block is thread-safe but not the access to its resource. That means, modifying the reference counter is an atomic operation and you have the guarantee that the resource will be deleted exactly once. This is the reason we will get with C++20 atomic smart pointers: `std::atomic_shared_ptr` and `std::atmic_weak_ptr`. Read the details in the proposal: Atomic Smart Pointers (http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2014/n4058.pdf).

Now to the more interesting part of C++20 and C++23. Pointers will be deprecated in C++20 and removed in C++23. Or to say it with three words: **N**o **N**ew **N**ew (NNN).

std::unique_ptr to our rescue

But hold, we have a dogma in C++: Don't pay for anything, you don't need. How can we program without a pointer? Just use an `std::unique_ptr`. A `std::unique_ptr` is by design as fast and as slim as a raw pointer but has a great benefit: it automatically manages its resource.

Only to remind you. Here is a simple performance test.

```
// all.cpp

#include <chrono>
#include <iostream>

static const long long numInt= 100000000;

int main(){

  auto start = std::chrono::system_clock::now();

  for ( long long i=0 ; i < numInt; ++i){
    int* tmp(new int(i));
    delete tmp;
    // std::shared_ptr<int> tmp(new int(i));
    // std::shared_ptr<int> tmp(std::make_shared<int>(
    // std::unique_ptr<int> tmp(new int(i));
    // std::unique_ptr<int> tmp(std::make_unique<int>(i));
  }

  std::chrono::duration<double> dur= std::chrono::system_clock::now() - start;
  std::cout << "time native: " << dur.count() << " seconds" << std::endl;

}
```


The program allocates and deletes 100.000.000 ints. I use a raw pointer, and `std::shared_ptr` and `std::unique_ptr` in two variations. I executed the program with and without maximum optimisation on Linux and Windows. Here are the numbers.

Compiler	Optimization	new	std::shared_ptr	std::make_shared	std::unique_ptr	std::make_unique
GCC	no	3.03	13.48	30.47	8.74	9.09
GCC	yes	3.03	6.42	3.24	3.07	3.04
cl.exe	no	8.79	25.17	18.75	11.94	13.00
cl.exe	yes	7.42	17.29	9.40	7.58	7.68

The numbers show it black on blue. The two variations of `std::unique_ptr` are on Linux and Windows as fast as the raw pointers. For the details of the numbers, read my previous post: Memory and Performance Overhead of Smart Pointers (https://www.modernescpp.com/index.php/memory-and-performance-overhead-of-smart-pointer).

Ownership semantic

Honestly, we use pointers and in particular raw pointers far too often. The question if you should use a pointer, boils down to one question: Who is the owner? Luckily, we can directly express our intention about ownership in code.

- **Local objects**. The C++ runtime as the owner automatically manages the lifetime of these resources. The same holds for global objects or members of a class. The guidelines call them scoped objects.
- **References**: I'm not the owner. I only borrowed the resource that cannot be empty.
- **Raw pointers**: I'm not the owner. I only borrowed the resource that can be can be empty. I must not delete the resource.
- **`std::unique_ptr`**: I'm the exclusive owner of the resource. I may explicitly release the resource.
- **`std::shared_ptr`**: I share the resource with other shared ptr. I may explicitly release my shared ownership.

- **`std::weak_ptr`**: I'm not the owner of the resource but I may become temporary the shared owner of the resource by using the method `std::weak_ptr::lock` (http://en.cppreference.com/w/cpp/memory/weak_ptr/lock).

We have only to change our practice in one out of six use-cases and we are fine with the next evolution step in C++.


(http://www.facebook.com/rainer.grimm.


(https://plus.google.com/109576694736


(https://www.linkedin.com/in/rainergrimm


(https://twitter.com/rainer_grimm)


(https://www.xing.com/profile/Rainer_Gr

Hunting (http://www.huntersspeak.com/)

What's next?

Sorry for the detour but the Jacksonville decision was worth a detour. [illegible] exposure will be about the rules to performance in the C++ Core Guidelines.

Further Information

The decision to the next pdf bundle is made. It will take me about a week to prepare the pdf-bundles. First, I have to look for typos. People already subscribed to my newsletter will get them automatically.

- English: Functional Features in C++ (https://www.modernescpp.com/index.php/which-pdf-bundle-should-i-provide-make-your-choice-2)
- German: Charakteristiken der funktionalen Programmierung (https://www.grimm-jaud.de/index.php/blog/welches-pdf-paeckchen-soll-ich-zusammenstellen-mache-dein-kreuz-3)

Thanks a lot to my Patreon Supporters (https://www.patreon.com/rainer_grimm): Matt Braun, Roman Postanciuc, Tobias Zindl, Marko, G Prvulovic, Reinhold Dröge, Abernitzke, Frank Grimm, Sakib, Broeserl, António Pina, Sergey Agafyin, Андрей Бурмистров, Jake, GS, Lawton Shoemake, Animus24, Jozo Leko, John Breland, espkk, Wolfgang Gärtner, Louis St-Amour, Venkat Nandam, Jose Francisco, Douglas Tinkham, Kuchlong Kuchlong, Robert Blanch, Truels Wissneth, Kris Kafka, Mario Luoni, Neil Wang, Friedrich Huber, lennonli, Pramod Tikare Muralidhara, Peter Ware, Tobi Heideman, Daniel Hufschläger, Red Trip, Alexander Schwarz, Tornike Porchxidze, Alessandro Pezzato, Evangelos Denaxas, Bob Perry, and Satish Vangipuram.

Thanks in particular to Jon Hess, Lakshman, Christian Wittenhorst, Sherhy Pyton, Dendi Suhubdy, Sudhakar Belagurusamy, Richard Sargeant, and Rusty Fleming.

My special thanks to Embarcadero (https://www.embarcadero.com/de/products/cbuilder)

(https://www.embarcadero.com/products/cbuilder)

Seminars

I'm happy to give online-seminars or face-to-face seminars world-wide. Please call me if you have any questions.

Bookable (Online)

German

- Clean Code mit modernem C++: (https://www.modernescpp.de/index.php/c/2-c/31-clean-code-mit-modernem-c)22.06.2021 - 24.06.2021
- C++20: (https://www.modernescpp.de/index.php/c/2-c/32-c-20) 8.2021 - 12.08.2021
- Embedded Programmierung mit modernem C++: (https://www.modernescpp.de/index.php/c/2-c/33-embedded-programmierung-mit-modernem-c20210421085111) 9.2021 - 23.09.2021


(http://www.facebook.com/rainer.grimm.

(https://plus.google.com/109576694736

(https://www.linkedin.com/in/rainergrimm

(https://twitter.com/rainer_grimm)

(https://www.xing.com/profile/Rainer_Gr
Hunting (http://www.huntersspeak.com/)

Standard Seminars (English/German)

Here is a compilation of my standard seminars. These seminars are designed and meant to give you a first orientation.

- C++ - The Core Language (https://www.modernescpp.net/index.php/c/plan/2-c/22)
- C++ - The Standard Library (https://www.modernescpp.net/index.php/c/plan/2-c/23)
- C++ - Compact (https://www.modernescpp.net/index.php/c/plan/2-c/23)
- C++11 and C++14 (https://www.modernescpp.net/index.php/c/plan/2-c/18)
- Concurrency with Modern C++ (https://www.modernescpp.net/index.php/c/plan/2-c/19)
- Design Patterns and Architecture Patterns with C++ (https://www.modernescpp.net/index.php/c/plan/2-c/21)
- Embedded Programming with Modern C++ (https://www.modernescpp.net/index.php/c/plan/2-c/17)
- Generic Programming (Templates) with C++ (https://www.modernescpp.net/index.php/c/plan/2-c/17)

New

- Clean Code with Modern C++ (https://www.modernescpp.net/index.php/c/plan/2-c/16)
- C++20 (https://www.modernescpp.net/index.php/c/plan/2-c/25)

Contact Me

- Phone: +49 152 31965939
- Mail: schulung@ModernesCpp.de (mailto:schulung@ModernesCpp.de) (https://www.modernescpp.com/<a href=)
- German Seminar Page: www.ModernesCpp.de (https://www.modernescpp.de/)
- English Seminar Page: www.ModernesCpp.net (http://www.ModernesCpp.net)

Modernes C++,

Tweet (http://twitter.com/share)

Like 14 Share

Comments (/index.php/component/jcomments/feed/com_jaggyblog/368)

#1 (/index.php/component/jaggyblog/no-new-new#comment-12994) **Rainer Grimm** 2018-04-02 06:00 **0**

Quoting Segey:

> LoL! 1. April glaube ich niemandem!

Stimmt! Glaube niemand am 1. April.

Quote

#2 (/index.php/component/jaggyblog/no-new-new#comment-18452) **chung cu hong linh** **0**
2018-07-18 01:27
I am in fact delighted to glance at this weblog
posts which carries tons of useful facts, thanks for providing these kinds of
data.

Quote

#3 (/index.php/component/jaggyblog/no-new-new#comment-206 **ộ chăn ra gối nệm** **0**
2018-08-20 05:57
Hello! Would you mind if I share your blog with my zynga group?
There's a lot of folks that I think would really enjoy your content.
Please let me know. Many thanks

Quote

#4 (/index.php/component/jaggyblog/no-new-new#comment-207 **ainer Grimm** 2018-08- **+1**
22 05:03
Quoting bộ chăn ra gối nệm:

> Hello! Would you mind if I share your blog with my zynga group?
> There's a lot of folks that I think would really enjoy your content.
> Please let me know. Many thanks

You can share my post. Just mention my blog and my name.

Quote

#5 (/index.php/component/jaggyblog/no-new-new#comment-23391) **Adrian** 2018-09-19 13:21 **+1**
Hello,

I appreciate a lot the fact that the commitee is working on ways to improve cpp safety, but I think it is an error to deprecate and than later remove pointers. I think a better solution would be to make the compiler accepts pointers only under a certain compilation flags, like, let's "--raw-pointers-on".
There will be ALWAYS high speed computation use case where the fast the best. ALWAYS.
If guy from, let's say, MISRA don't like pointers, just tell them to put this flag on their rules.

Regards,
Adrian

Quote

#6 (/index.php/component/jaggyblog/no-new-new#comment-38912) **Ivan** 2019-03-20 21:47 **0**
Keep on working, great job!

Quote

#7 (/index.php/component/jaggyblog/no-new-new#comment-46196) **ozeitu** 2019-05-19 04:01 **+5**
"C++ Will No Longer Have Pointers" is an april fools' joke...

Quote

#8 (/index.php/component/jaggyblog/no-new-new#comment-67812) **Roland Hughes** 2019- **+1**
12-04 14:35
Stumbling across this site was a pleasant surprise. Removal of pointers is going to break an ocean of code. It will also mean C++ can no longer be used for low level development, particularly where hardware must be accessed via address and data transferred to it via memcpy(). It will also devastate one of the most popular cross platform toolkits out there.
https://doc.qt.io/qt-5/qtwidgets-widgets-analogclock-example.html

Quote

#9 (/index.php/component/jaggyblog/no-new-new#comment-77866) **Brian** 2020-07-12 06:24 **+5**
You really should edit with a note about the April fool's joke. Google is putting this in top few results for "Modern C++ Pointers"

Quote

#10 (/index.php/component/jaggyblog/no-new-new#comment-77978) **Joon** 2021-05-16 16:18 **0**
Oh my god this was such a scare stumbling upon on definitely not April 1st

Quote

Refresh comments list

RSS feed for comments to this post (/index.php/component/jcomments/feed/com_jaggyblog/368)

Add comment

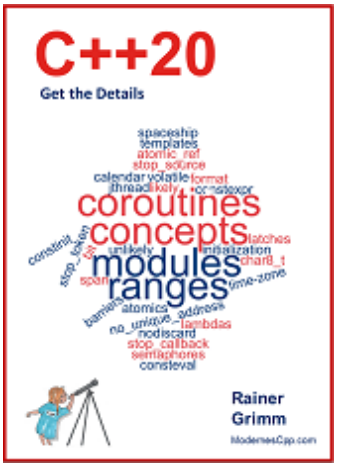
(https://www.modernescpp.net/)

Contact

rainer@grimm-jaud.de (/index.php/rainer-grimm)

Impressum (/index.php/impressum)

My Newest E-Books



(https://leanpub.com/c20)

(http://www.facebook.com/rainer.grimm.

(https://plus.google.com/109576694736

(https://www.linkedin.com/in/rainergrimn

(https://twitter.com/rainer_grimm)

(https://www.xing.com/profile/Rainer_Gr

(https://www.patreon.com/rainer_grimm)

Subscribe to the newsletter (+ pdf bundle)

Name

Email

Subscribe

>> Start here <<

(/index.php/der-einstieg-in-modernes-c)

>> Visual Tour <<

(https://youtu.be/hrXoVSi0O28)

Categories

- C++17 (/index.php/category/c-17)
- C++20 (/index.php/category/c-20)
- C++ Core Guidelines (/index.php/category/modern-c)
- C++ Insights (/index.php/category/cppinsight)
- Embedded (/index.php/category/embedded)
- Functional (/index.php/category/functional)
- Multithreading (/index.php/category/multithreading)
- Multithreading - Application (/index.php/category/multithreading-application)
- Multithreading - C++17 and C++20 (/index.php/category/multithreading-c-17-and-c-20)
- Multithreading - Memory Model (/index.php/category/multithreading-memory-

(https://leanpub.com/concurrencywithmodernc)

(https://leanpub.com/cpplibrary)

Course: Modern C++ Concurrency in Practice

(https://www.educative.io/courses/modern-cpp-concurrency-in-practice-get-the-most-out-of-any-machine?authorName=Rainer+Grimm)

Course: C++ Standard Library including C++14 & C++17

model)
- News (/index.php/category/news)
- Overview (/index.php/category/ueberblick)
- Pdf bundles (/index.php/category/pdf-bundle)
- Review (/index.php/category/review)
- Templa index.php/category/templates)

(http://www.facebook.com/rainer.grimm.

(https://plus.google.com/109576694736

(https://www.linkedin.com/in/rainergrimm

(https://twitter.com/rainer_grimm)

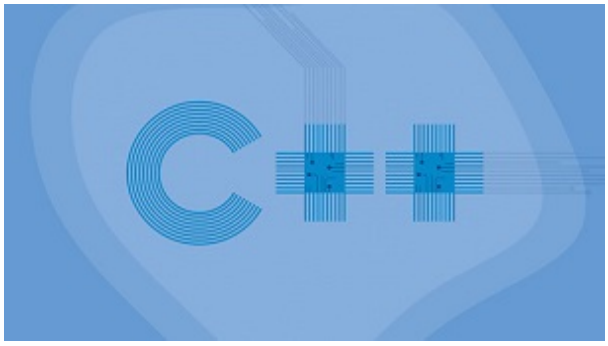
(https://www.xing.com/profile/Rainer_Gr

All tags

acquire-release (https://www.modernescpp.com/index.php/tag/acquire-release) relaxed semantik (https://www.modernescpp.com/index.php/tag/relaxed-semantik) arithmetic (/index.php/tag/arithmetic) associative containers (/index.php/tag/associative-container) async (/index.php/tag/async) atomics (/index.php/tag/atomics) atomic_thread_fence Hunting (http://www.huntersspeak.com/) (/index.php/tag/atomic-thread-fence) auto (/index.php/tag/auto) bit manipulation (/index.php/tag/bit-manipulation) C (/index.php/tag/c) C++17 (/index.php/tag/c-17) C++20 (/index.php/tag/c-20) class hierarchies (/index.php/tag/class-hierarchies) classes (/index.php/tag/classes) concepts (/index.php/tag/concepts) condition variables (/index.php/tag/condition-variable) constexpr (/index.php/tag/constexpr) contracts (/index.php/tag/contracts) conversions (/index.php/tag/conversions) coroutines (/index.php/tag/coroutines) CppMem (/index.php/tag/cppmem) declarations (/index.php/tag/declarations) decltype (/index.php/tag/decltype) enum (/index.php/tag/enum) error handling (/index.php/tag/error-handling) exceptions (/index.php/tag/exceptions) expressions (/index.php/tag/expressions) final (/index.php/tag/final) finally (/index.php/tag/finally) functions (/index.php/tag/functions) Guideline Support Library (/index.php/tag/guideline-support-library) if (/index.php/tag/if) initialisations (/index.php/tag/initialisations) inline (/index.php/tag/inline) interfaces (/index.php/tag/interfaces) lambdas (/index.php/tag/lambdas) lock (/index.php/tag/lock) lock-free (/index.php/tag/lock-free) memory (/index.php/tag/memory) memory_order_consume (/index.php/tag/memory-order-consume) modules (/index.php/tag/modules) move (/index.php/tag/move) mutex (/index.php/tag/mutex) new/delete (/index.php/tag/new-delete) noexcept (/index.php/tag/noexcept) nullptr (/index.php/tag/nullptr) Ongoing Optimization (/index.php/tag/ongoingoptimization) overloading (/index.php/tag/overloading) override (/index.php/tag/override) performance (/index.php/tag/performance) pointers (/index.php/tag/pointers) ranges library (/index.php/tag/ranges-library) relaxed semantic (/index.php/tag/relaxed-semantik) semaphores (/index.php/tag/semaphores) sequential consistency (/index.php/tag/sequential-consistency) shared_ptr (/index.php/tag/shared-ptr) singleton (/index.php/tag/singleton) smart pointers

(https://www.educative.io/collection/10370001/5712008708096000?authorName=Rainer%20Grimm)

Course: Embedded Programming with Modern C++



(https://www.educative.io/courses/embedded-programming-with-cpp)

Course: Generic Programming (Templates)

(https://www.educative.io/courses/generic-templates-in-cpp)

Course: C++ Fundamentals for Professionals

(https://www.educative.io/courses/cpp-

(/index.php/tag/smart-pointers) source files (/index.php/tag/source-files) spaceship (/index.php/tag/spaceship) statements (/index.php/tag/statements) static (/index.php/tag/static) static_assert (/index.php/tag/static-assert) switch (/index.php/tag/switch) tasks (/index.php/tag/tasks) template (/index.php/tag/template-metaprogramming) templates (/index.php/tag/templates) ThreadSanitizer (/index.php/tag/threadsanitizer) thread_local (/index.php/tag/thread-local) time (/index.php/tag/time) type-erasure (/index.php/tag/type-erasure) type-traits (/index.php/tag/type-traits) unique_ptr (/index.php/tag/unique-ptr) user-defined literals (/index.php/tag/benutzerdefinierte-literale) vector (/index.php/tag/vector) volatile (/index.php/tag/volatile) weak_ptr (/index.php/tag/weak-ptr)

Blog archive

Source Code

GitHub (https://github.com/RainerGrimm/ModernesCppSource

Most Popular Posts

- Thread-Safe Initialization of a Singleton (266517 hits) (https://www.modernescpp.com/index.php/thread-safe-initialization-of-a-singleton)
- C++ Core Guidelines: Passing Smart Pointers (241190 hits) (https://www.modernescpp.com/index.php/c-core-guidelines-passing-smart-pointer)
- C++17 - Avoid Copying with std::string_view (218716 hits) (https://www.modernescpp.com/index.php/c-17-avoid-copying-with-std-string-view)
- C++ Core Guidelines: Be Aware of the Traps of Condition Variables (207322 hits) (https://www.modernescpp.com/index.php/c-core-guidelines-be-aware-of-the-traps-of-condition-variables)
- C++20: Structure Modules (201000 hits) (https://www.modernescpp.com/index.php/c-20-divide-modules)

fundamentals-for-professionals)

More Profiles

Training, coaching, and technology consulting (https://www.modernescpp.net/)

Visitors

Today	4982
Yesterday	7550
Week	27184
Month	28043
All	6460124

Currently are 196 guests and no members online

Kubik-Rubik Joomla! Extensions (https://kubik-rubik.de/)

(http://www.facebook.com/rainergrimm.

(https://plus.google.com/109576694736

(https://www.linkedin.com/in/rainergrimn

(https://twitter.com/rainer_grimm)

(https://www.xing.com/profile/Rainer_Gr

Hunting (http://www.hunterspeak.com/)

Latest comments

Performance Comparison of Condition Variables and Atomics in C++20 (/index.php/component/jaggyblog/perfor of-condition-variables-and-atomics-in-c-20)

Alex

I just installed gcc 11.1 to try some new c++20 features. There is bug in pong method , it should be ...

Read more... (/index.php/component/jaggyblog/performancecor of-condition-variables-and-atomics-in-c-20#comment-77980)

C++ is Lazy: CRTP (/index.php/component/jaggyblog/c-is-still-lazy)

Daniel T

Lines 18 and 19 are unused

Read more... (/index.php/component/jaggyblog/c-is-still-lazy#comment-77979)

No New New: Raw Pointers Removed from C++ (/index.php/component/jaggyblog/no-new-new)

Joon

Oh my god this was such a scare stumbling upon on definitely not April 1st

Read more... (/index.php/component/jaggyblog/no-new-new#comment-77978)

Templates - First Steps (/index.php/component/jaggyblog/templ get-insight)

Pramod Tikare

Nice article. I have got 2 feedbacks. 1. It makes sense to add line numbers to your code snippet ...

Read more... (/index.php/component/jaggyblog/template-get-insight#comment-77977)

Functi Templates (/inde p/component/jaggyblog/funeti templ)

Kiran

Hi Rai Thanks for writing such awesome content. Two description lines I found, might need ...

Read more... (/index.php/component/jaggyblog/function-templates#comment-77976)