

Learning (and Teaching) Modern C++ Challenges and Resources



Intro

Who?

- Arne Mertz
- C++ for ~ 10 years
 - Except the last
- Blogging since 2015
 - About modern C++
 - Clean code
 - Testing
 - ...

Verse 1

Modern C++?

So what is "Modern C++"?



Meeting C++

@meetingcpp

So what is "Modern C++"?



Meeting C++

@meetingcpp

So what is "Modern C++"?

13% new standards + boost

52% new standards \geq C++11

21% ask Alexandrescu

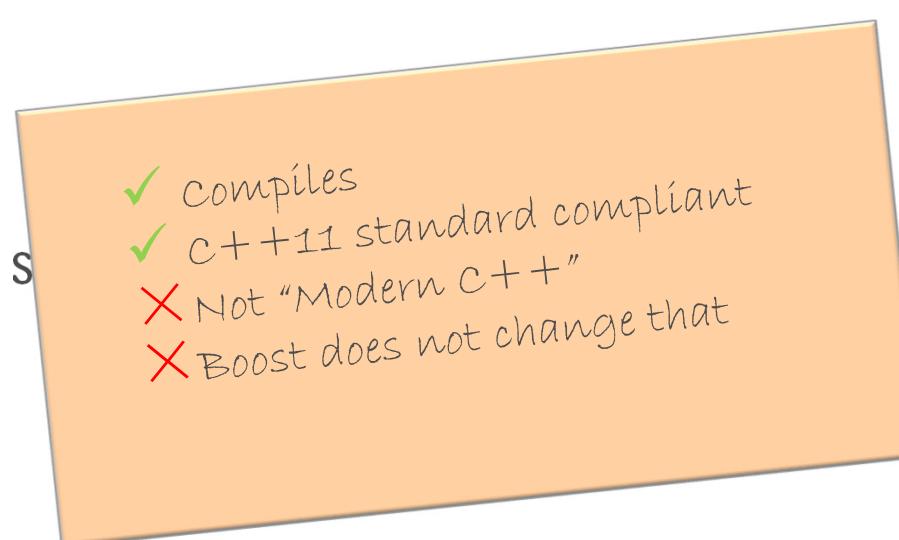
14% using C++ to its fullest

375 votes • Final results

The (new) standards define what is possible

Not what is good

- Free-standing new and delete
- void*
- int array[]
- “C with classes”
- Everything into classes, Java s



- ✓ Compiles
- ✓ C++11 standard compliant
- ✗ Not “Modern C++”
- ✗ Boost does not change that



Meeting C++

@meetingcpp

So what is "Modern C++"?

~~10%~~ new standards + boost

~~52%~~ new standards \geq C++11

21% ask Alexandrescu

14% using C++ to its fullest

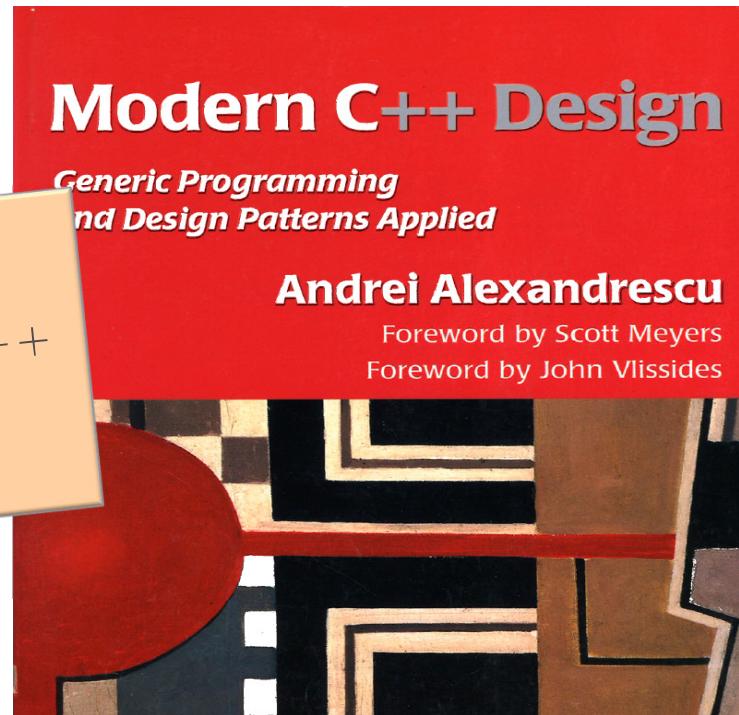
375 votes • Final results

“Modern C++ Design”

Andrei Alexandrescu

- From 2001 – C++ has evolved since then
- Very focused on templates
 - Policy based class design
 - Allocators
 - Type lists
 - Compile time assertions

- Still a good read
- Influenced our thinking about C++
- But not “Modern C++” anymore



My understanding is that the book "Modern C++ Design" coined the term "modern C++". The term refers to a template-intensive, generic style of writing code.

Andrei Alexandrescu

Modern art

The image shows a screenshot of a Wikipedia page comparing 'Modern art' and 'Contemporary art'. The page has a large orange banner at the top with the text 'Contemporary art' in black. Below the banner, there is a comparison table with two columns. The left column is titled 'Modern art' and the right column is titled 'Contemporary art'. The text in the 'Modern art' column is highlighted with a red box and contains the text '1860s to the 1970s,'.

Modern art	Contemporary art
extending roughly from the 1860s to the 1970s, style and philosophy of the art period. is usually associated with art in which the traditional canons of beauty and taste have been thrown aside in a spirit of experimentation. ^[2] Modern artists experimented with new ways of seeing and with fresh ideas about the nature of materials and functions of art. A tendency away from the narrative, which was characteristic for the traditional arts, toward abstraction is characteristic of much modern art.	extending roughly from the 1970s to the present, style and philosophy of the art period. is usually associated with art in which the traditional canons of beauty and taste have been thrown aside in a spirit of experimentation. ^[2] Contemporary artists experimented with new ways of seeing and with fresh ideas about the nature of materials and functions of art. A tendency away from the narrative, which was characteristic for the traditional arts, toward abstraction is characteristic of much contemporary art.

accu 2017

Functional C++ For Fun And Profit

Phil Nash

C++11 gave us lambda
boost::lambda) - so
more to functional
I'd even argue we
do functional pro

the language for the first time (if you ignore
language now, right? There's a bit

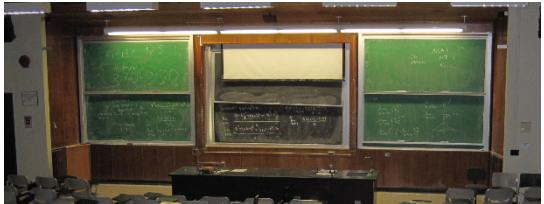
Modern C++ Design reloaded

Odin Holmes

Many of the concepts presented in Andrei Alexandrescu's book "Modern
C++ Design" were arguably ahead of their time. Many of the concepts
could not be effectively or cleanly expressed due to a lack of language
features and the user was often exposed to too much complexity.

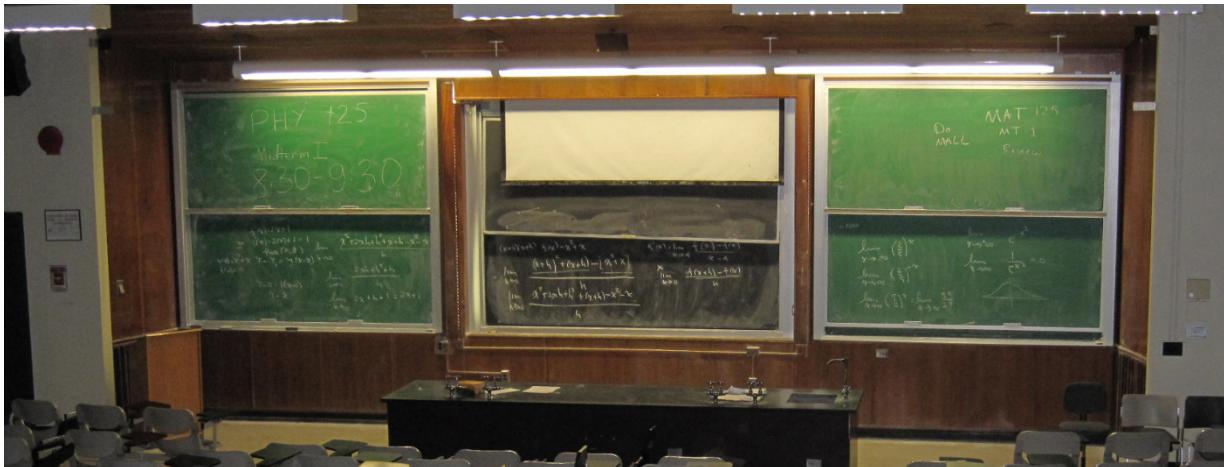
Now 15 years later we have many more tools. Now Alexandrescu's as
well as similar statically linking yet powerful and generic design patterns
are proving invaluable in resource constrained and low latency fields.

In this talk I will share my experience creating "post modern" C++
abstractions for bare metal hard realtime systems as well as my work on
the tools needed to provide sleek and sexy public interfaces
libraries such as various improv...



Learning (and Teaching) the right mix of Modern, Postmodern, Functional, Contemporary, classical, ... C++?

 @arne_mertz



Learning (and Teaching) the C++ We Use these Days

 @arne_mertz

Modern



Definition of *modern* in English:

modern

ADJECTIVE

- 1 Relating to the present or recent times as opposed to the remote past.

'the pace of modern life'

'modern European history'

- 1.1 Characterized by or using the most up-to-date techniques, ideas, or equipment.

'they do not have modern weapons'

- 1.2 **[attributive]** Denoting the form of a language that is currently used, as opposed to any earlier form.

'modern German'

- 1.3 **[attributive]** Denoting a current or recent style or trend in art, architecture, or other cultural activity marked by a significant departure from traditional styles and values.

'Matisse's contribution to modern art'



Meeting C++

@meetingcpp

So what is "Modern C++"?

~~13%~~ new standards + boost

~~52%~~ new standards \geq C++11

~~21%~~ ask Alexandrescu

14% using C++ to its fullest ✓

375 votes • Final results

Modern C++ is...

... what we consider best practice *today*

- Use the features of the current (and upcoming) standards sensibly
 - E.g. use auto, but not *always*
- But also the old
 - E.g. RAII is *still* modern
- Combine to get the best out of the language



Bridge

“So write a book!”



Your path through the pitfalls of the ancient



C++ - the good parts

The Safe Guide

O RLY?

Community

The Good Parts

Foreword

“When I was a young journeyman programmer, I would learn about every feature of the languages I was using, and I would attempt to use all of those features when I wrote. I suppose it was a way of showing off, and I suppose it worked because I was the guy you went to if you wanted to know how to use a particular feature.

Eventually I figured out that some of those features were more trouble than they were worth.[...] Most programming languages contain good and bad parts. I discovered that I could be a better programmer by using only the good parts and avoiding the bad parts. After all, how can you build something good out of bad parts?

It is rarely possible for standard committees to remove imperfections from a language because doing so would cause the breakage of all the bad programs that depend on those bad parts. [...]

But *you* have the power to define your own subset. You can write better programs by relying exclusively on the good parts.”

Douglas Crockford – “JavaScript: The Good Parts”

"Within C++ is a smaller, simpler, safer language struggling to get out."

Bjarne Stroustrup

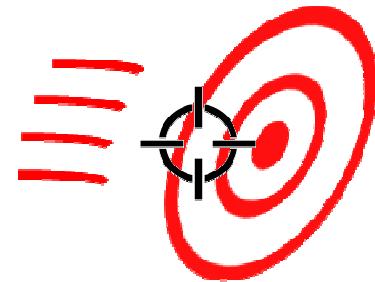
Verse 2

Challenges

Modern C++ is...

... a moving Target

- Frequently new features
 - New standards currently every ~3 years
 - Plus TS
- But besides that, our *understanding* of the language continually changes
 - E.g. “universal references”
 - E.g. “Almost Always Auto”



```
Widget w{"myWidget"};  
  
auto w = Widget{"myWidget"};
```

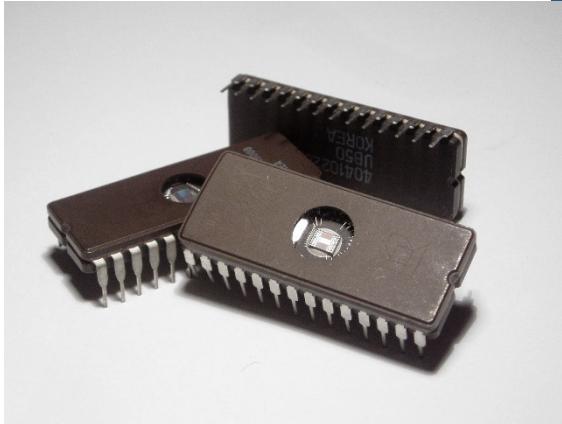
Modern C++ is...

... a moving target

```
template <class K, class V>
void printMap(std::map<K, V> const& m) {
    for(std::map<K, V>::iterator it = m.begin(); it != m.end(); ++it) {
        K const& key = it->first;
        V const& value = it->second;
        std::cout << "(" << key << ": " << value << ")\n";
    }
}

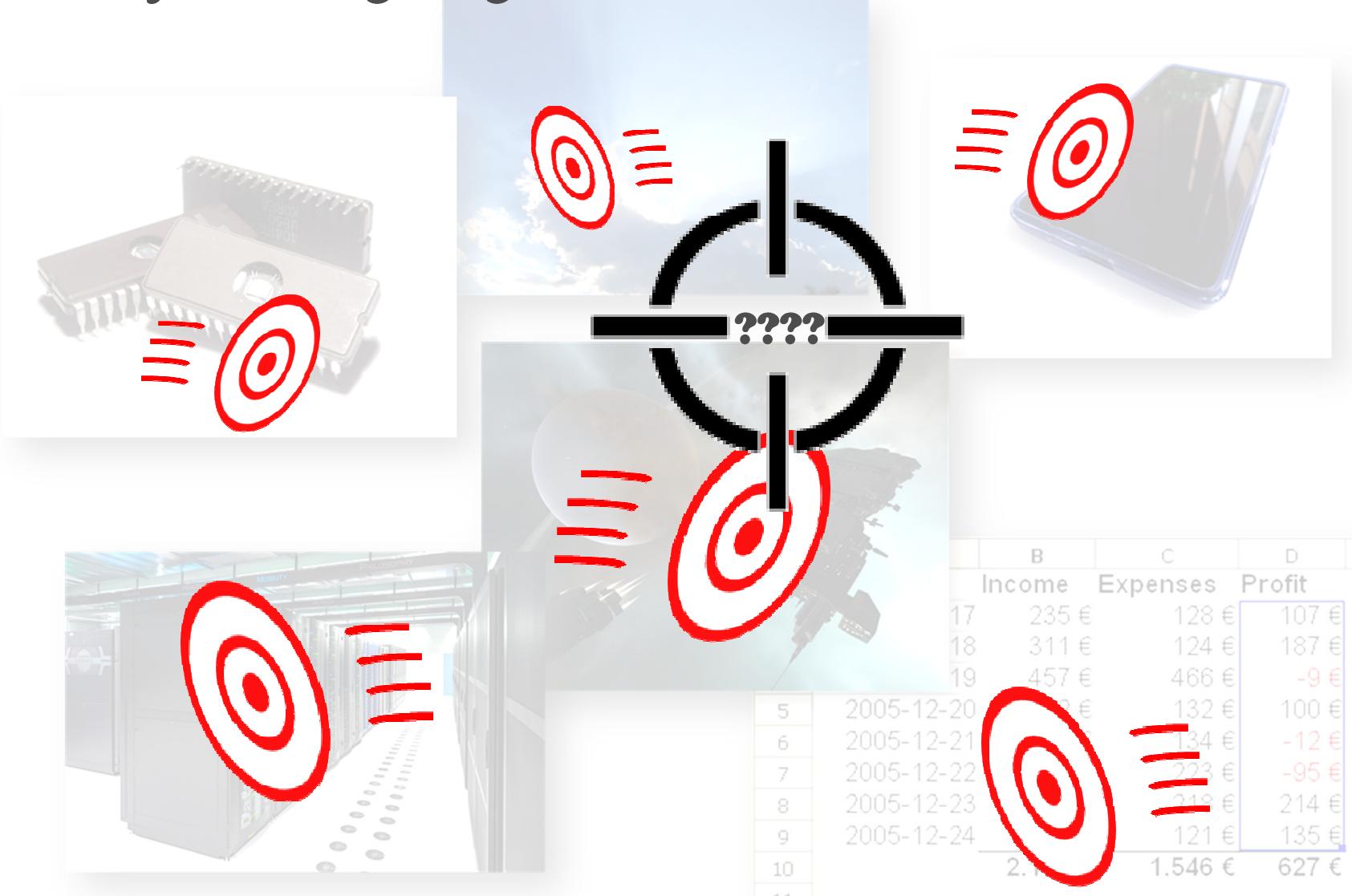
template <class Map>
void printMap(Map const& m) {
    for(auto const& e : m) {
        auto const& key = e.first;
        auto const& value = e.second;
        std::cout << "(" << key << ": " << value << ")\n";
    }
}

template <class Map>
void printMap(Map const& m) {
    for(auto const& [key, value] : m) {
        std::cout << "(" << key << ": " << value << ")\n";
    }
}
```



	B	C	D
	Income	Expenses	Profit
17	235 €	128 €	107 €
18	311 €	124 €	187 €
19	457 €	466 €	-9 €
5	2005-12-20	232 €	132 €
6	2005-12-21	122 €	134 €
7	2005-12-22	128 €	223 €
8	2005-12-23	432 €	218 €
9	2005-12-24	256 €	121 €
10		2.173 €	1.546 €
			627 €

Many moving targets...



Many moving targets...

... e.g. sales pitch for RAII

- Usual examples:
 - `unique_ptr` (memory)
 - `lock_guard` (mutexes)
 - `fstream` (files)
- Usual argument: Exception safety



Additional challenge

... for teachers

- We need to keep up
 - In addition to our normal jobs
- Constant (re-)learning required
- Have to explain that things change to students

Additional challenge

... for teachers

- We have learned modern C++ to be
 - Old C++ (or even C)
 - plus changes
- Modern C++ != layered evolution

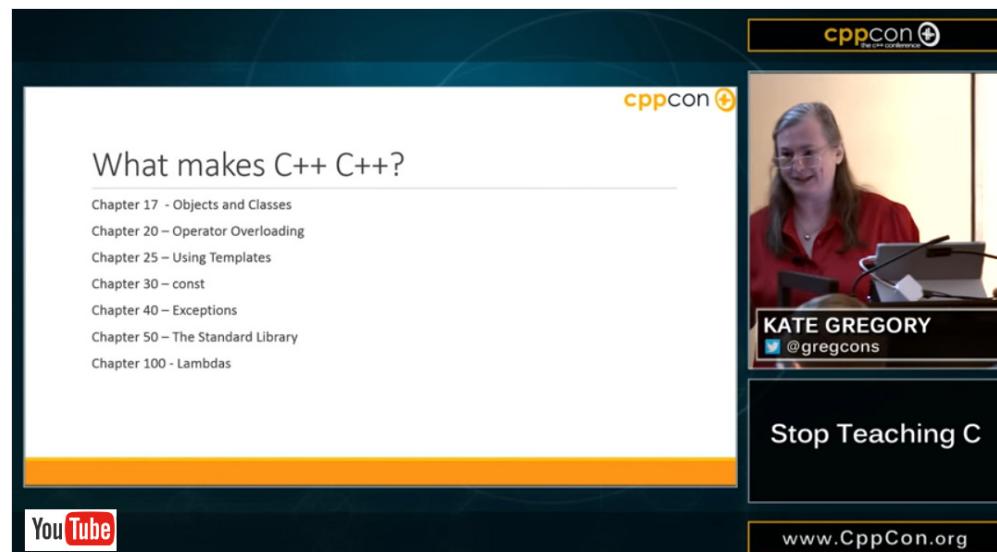
Additional challenge

... for teachers

- What we see and learn first sticks best
- Teach straight to the point instead of retelling our personal history

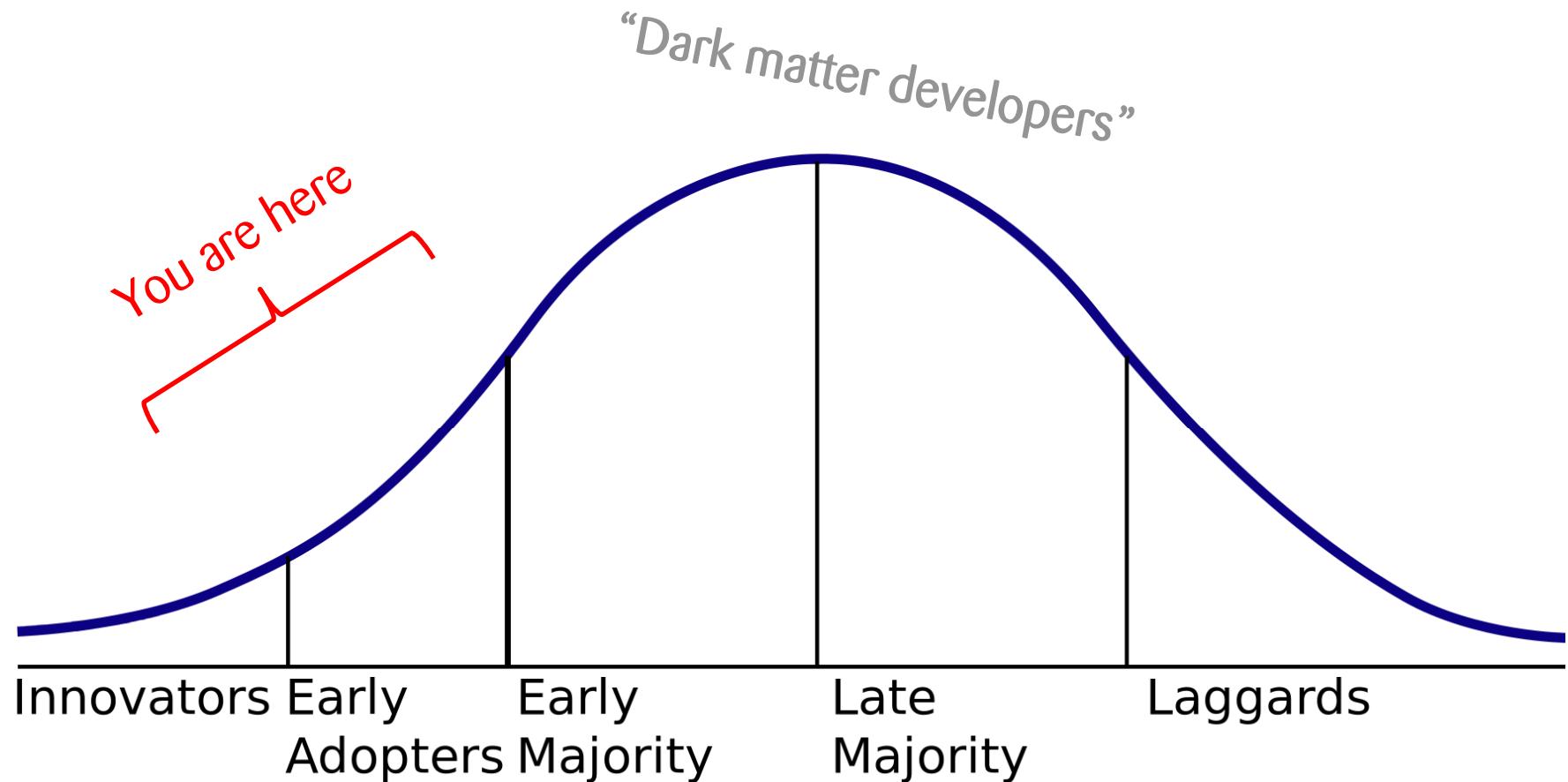
Example: start with `make_unique` & `unique_ptr` instead of `new/delete` → smart pointers

→ Kate Gregory:
“Stop Teaching C”
@ CppCon 2015



Adoption takes time

Not everyone uses the newest stuff...
... in fact, most probably don't.



Verse 3

Resources



Arne Mertz

@arne_mertz

When learning C++, what was your major source of knowledge?



Arne Mertz

@arne_mertz

When learning C++, what was your major source of knowledge?

6% Teacher/Professor

56% Books

31% Online Tutorials & Blogs

7% Other (please specify)

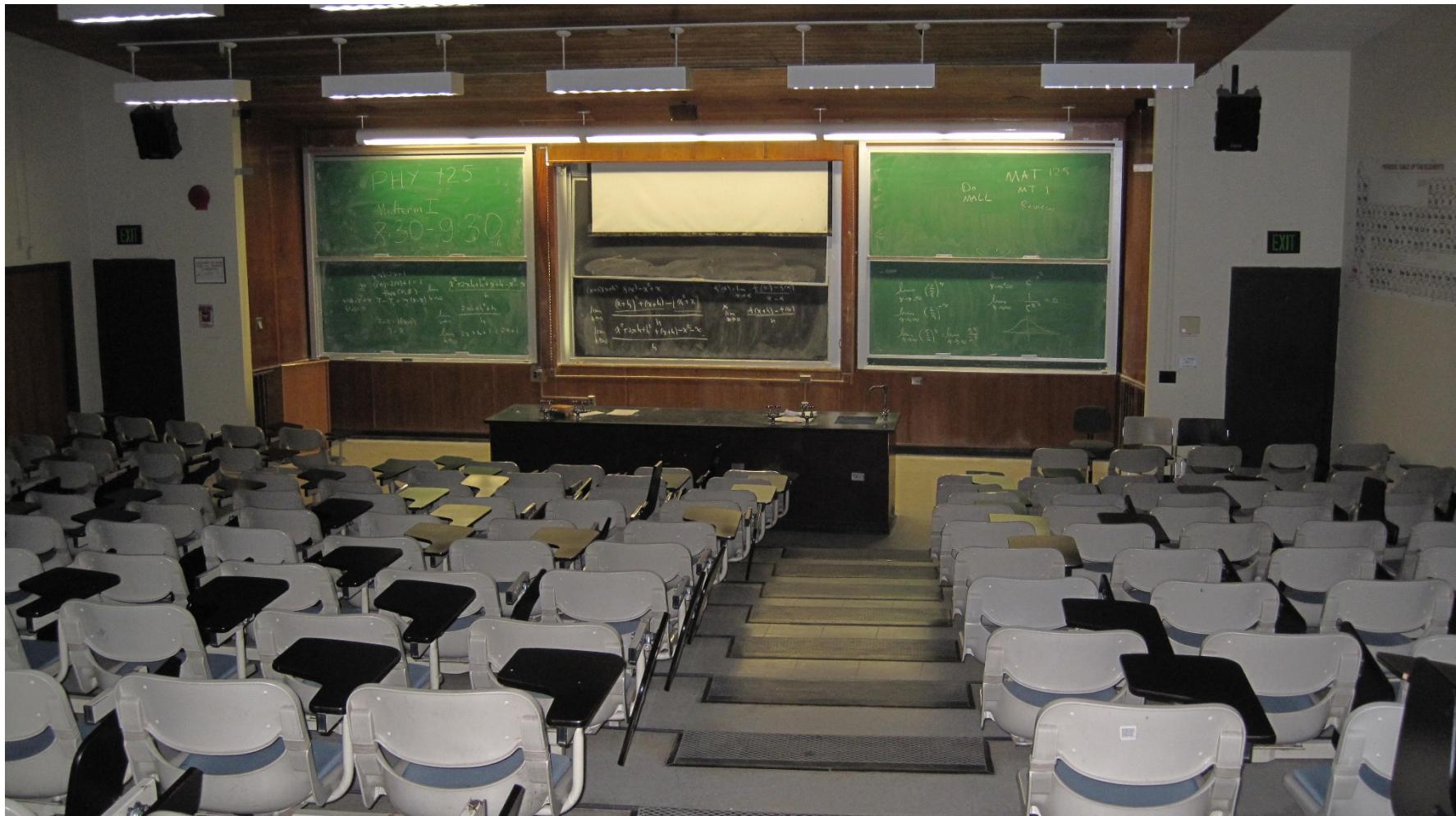
469 votes • Final results

Other:

- + Colleagues
- + Reading code
- + Magazines
- + Usenet
- + Books!
- + Conference videos

Class room...

... and teachers/professors



Class room...

... and teachers/professors

- Some teachers are enthusiastic and love to teach
 - Others don't but *have* to
 - Prepare class once
 - Repeat for N years
- fresh legacy C++ students

Class room...

Stop using Turbo C++: That is now stupid

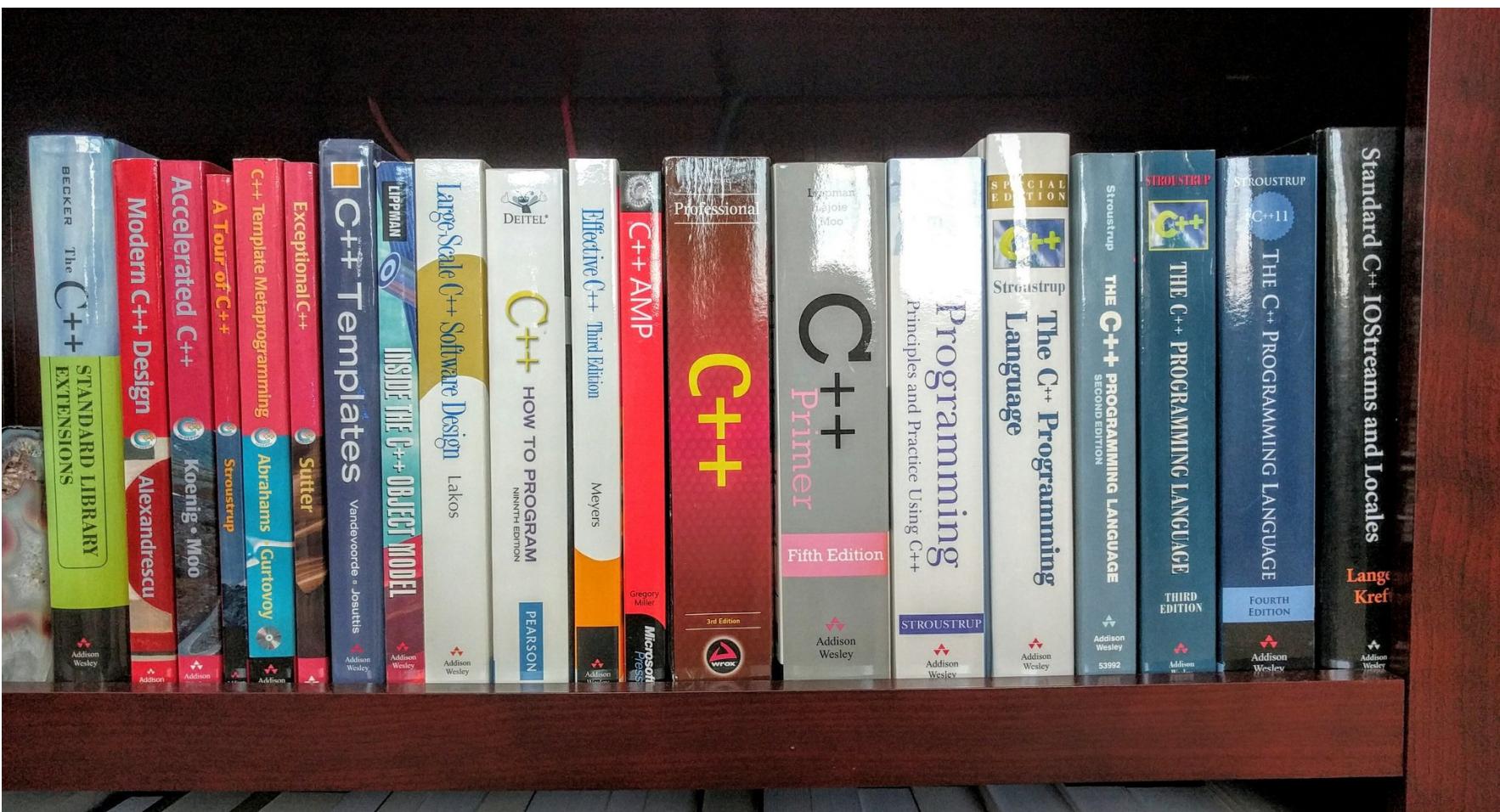
Sep 09, 2015 | by Saurabh Tripathi | in Opinions

Saurabh Tripathi → Arne Mertz • a year ago

Thanks for your comment, I completely agree that Turbo C++ is old and it should not be used. In most of the engineering collages in India, students are forced to use Turbo C++ in classrooms. I think you can understand why the post still makes sense for Indian students.

Reply Share

Books



Books

Pros:

- Consistency, didactic line of thought
- Usually reviewed (but not always)

Cons:

- Last longer than the validity of the content
- Are often promoted to make money – not because they are good
- Errors are hard to fix
- Usually not free of charge (*this does matter!*)

Books

C++ <iostream.h> Error



I am at the absolutely newest level of new when it comes to C++. It may seem like a noob mistake, but I think I'm missing something with my first program, "Hello World!".

0



I'm running from Ubuntu (not sure if this is any different from working with Windows), and I'm using a book called *Teach Yourself C++ in 21 Days*.



The code I'm resembling looks exactly like this:

```
#include <iostream.h>
int main()
{
    cout <<"Hello World!\n";
    return 0;
}
```

I have this exactly in my text editor, but I keep getting greeted by the same error whenever I try to compile it!

first.cpp:2:22: fatal error: iostream.h: No such file or directory compilation terminated.

I'm pretty distressed as this is literally the first step in my coding career! I'm not sure if ubuntu needs to be treated differently than Windows (which is what the book is using as reference).

Help!

asked Jul 13 '13 at 20:16

Blogs & Tutorials

Meeting C++ Blogroll 79
2016-11-11 10:19 by Jens Weller

Blogroll No. 79 - 11. November

- [agilecxx - Templates and binary bloat](#)
- [Aras - Interview questions](#)
- [Arne Mertz - Modern C++ Features – Variadic Templates](#)
- [Bartek's Coding Blog - Variadic Templates and a Factory Function](#)
- [Bits of Bytes - How to contribute to an open source project on Github](#)
- [Bitwise Bytes - How to use databases in your application with SQLite and Qt](#)
- [C++ Island - sizeof_ When The Whole is Greater Than The Sum Of Its Parts](#)
- [C++ Truths - Dependently-typed Curried printf](#)
- [Clion - Clion 2016.3 Release Candidate](#)
- [CPP Rendering - Vulkan Memory Management : How to write your own allocator](#)
- [CPP Rendering - Barriers in Vulkan : They are not that difficult](#)
- [Dimitar Mirchev - C++ tips, 2016 Week 44 \(31-Oct - 6-Nov-2016\)](#)
- [Italian C++ - C++ Day 2016](#)
- [Ivan Cukic - Functional Programming in C++ book, and the promo discount codes](#)
- [Jacko's C++ Blog - Python Style printf for C++ with printpp](#)
- [jemalloc - 4.3.0](#)
- [jemalloc - 4.3.1](#)
- [Josh Habermann - Introducing Bloaty McBloatface: a size profiler for binaries](#)
- [Kenny Kerr - C++/WinRT: Working with Implementations](#)
- [Krister Walfridsson - Inlining — shared libraries are special](#)
- [Krister Walfridsson - "missing" optimizations — constant address comparison](#)
- [kukuruku - Asynchronous Programming Part 2: Teleportation through Portals](#)
- [Marius Bancila - My book on modern C++ programming](#)
- [Meeting C++ - Collaborative Online C++ Compiler?](#)
- [Meeting C++ Blogroll - Meeting C++ Blogroll 78](#)
- [Meeting C++ News - PPQ = Pizza & Pasta and a Quiz!](#)
- [Meeting C++ News - Meeting C++ 2016 is sold out!](#)
- [Modernes C++ - The null pointer constant nullptr](#)
- [Modernes C++ - inline](#)
- [Modernes C++ - Constant expressions with constexpr](#)
- [NVIDIA DevBlog - New Compiler Features in CUDA 8](#)
- [pzemtsov - A bug story: data alignment on x86](#)
- [Qt Blog - Qt on the NVIDIA Jetson TX1 – Device Creation Style](#)
- [Qt Blog - Qt Visual Studio Tools 2.0 Released](#)
- [Qt Blog - Over-the-Air Updates, Part 3: Repository Configuration and Handling](#)
- [Rainer Grimm - Reine Funktionen](#)
- [Rambling Comments - Practical Testing: 34 - Potential reentrant locking deadlock](#)
- [Rambling Comments - C++ Tools - Some thoughts on JetBrains ReSharper C++](#)
- [Rambling Comments - C++ Tools - JetBrains ReSharper C++ is slowly winning me over](#)
- [Rambling Comments - Practical Testing: 36 - Timeout handle wrap](#)
- [Rebecca Fernandez - Object pools, variadic templates and reference forwarding](#)
- [SanSS - Database transaction handling in C++ systems](#)
- [The Old New Thing - How do I programmatically add a folder to my Documents library?](#)
- [videocortex - Terminators](#)
- [Visual Studio Blog - Developing Linux C++ applications with Azure Docker containers](#)
- [Visual Studio Blog - Visual C++ docs: the future is... soon!](#)
- [zverovich.net - Reducing printf call overhead with variadic templates](#)

Blogs & Tutorials

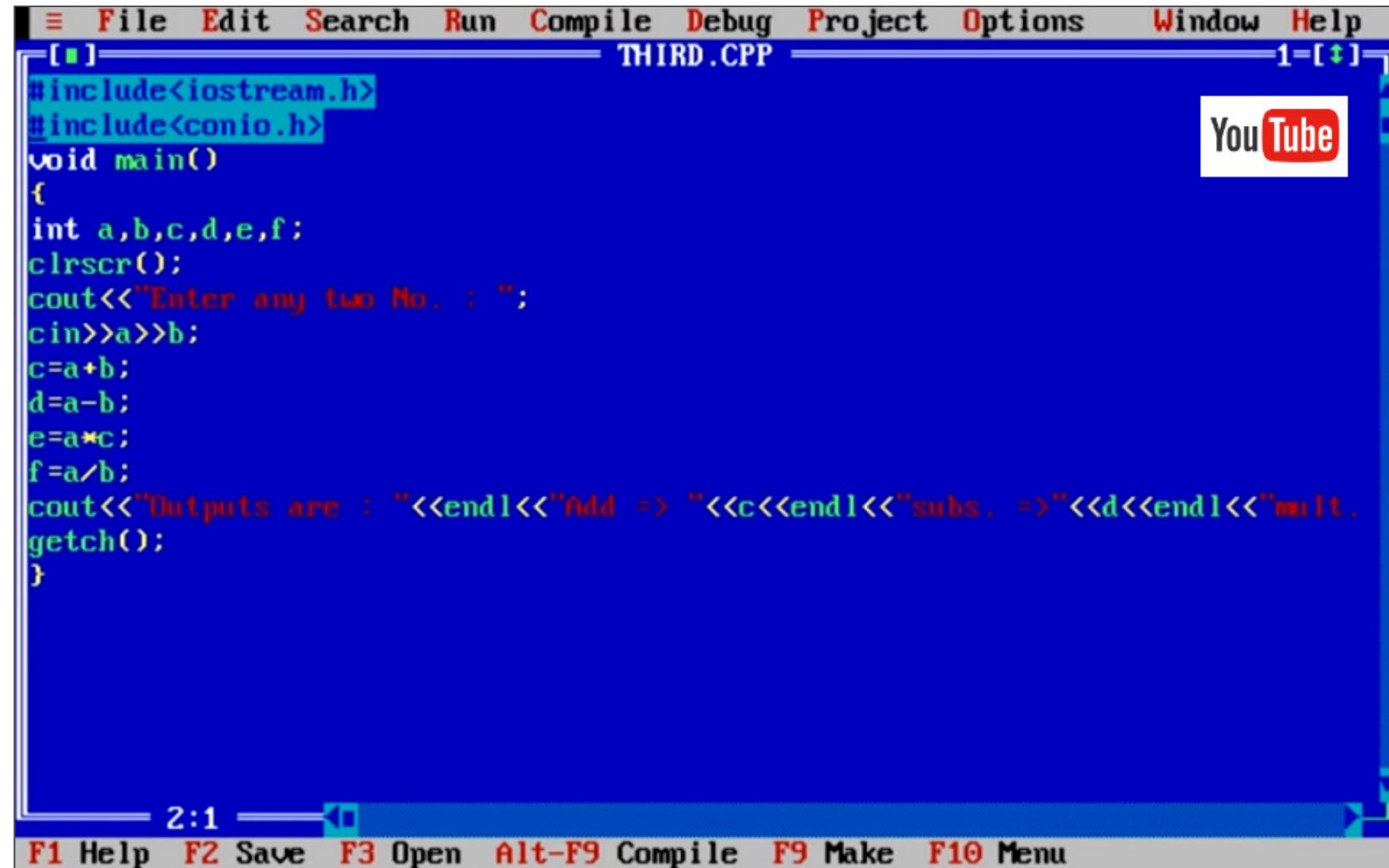
Pros:

- Usually free of charge
- Authors can always fix things or take them offline

Cons:

- But we usually don't
- Hardly any review
- Less coherent bits and pieces

Blogs & Tutorials



The screenshot shows a Microsoft Visual Studio IDE window titled "THIRD.CPP". The menu bar includes File, Edit, Search, Run, Compile, Debug, Project, Options, Window, and Help. The code editor contains the following C++ code:

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a,b,c,d,e,f;
clrscr();
cout<<"Enter any two No. : ";
cin>>a>>b;
c=a+b;
d=a-b;
e=a*c;
f=a/b;
cout<<"Outputs are : "<<endl<<"Add => "<<c<<endl<<"subs. =>"<<d<<endl<<"mult.
getch();
}
```

The status bar at the bottom shows "2:1" and the keyboard shortcut "F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu". A YouTube logo watermark is visible in the top right corner of the IDE window.

Blogs & Tutorials

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
Void main()
{
    clrscr();
    Char a[20];
    Cout<<"enter string "<<endl;
    Cin>>a;
    Strlwr(a); // strupr(a) for uppercase letter as output
    Cout<< string in lower case letters :"<<a;
    Getch();
}
```

String handling functions in C++

In c++ we have many string handling functions, which are used for comparing ,reversing ,and joining or addition of string and much more. These all string handling functions contained in "string.h" header file. so whenever we have to perform any related operation then we have to include string.h header file in our program.

In this article we are going to learn about following string handling functions.

.Strlen() – used to find the length of string

.Strcat() – used to add two strings.

.Strcpy() – used to copy string

.Strrev() – used to reverse a string

.Strlwr() – used to change letters to lowercase letters



Author:

is young indian boy who enjoy coding and passionate to make , break & Develop New Codes .



Twitter



Facebook



Google+

Colleagues & reading Code



Colleagues & reading Code

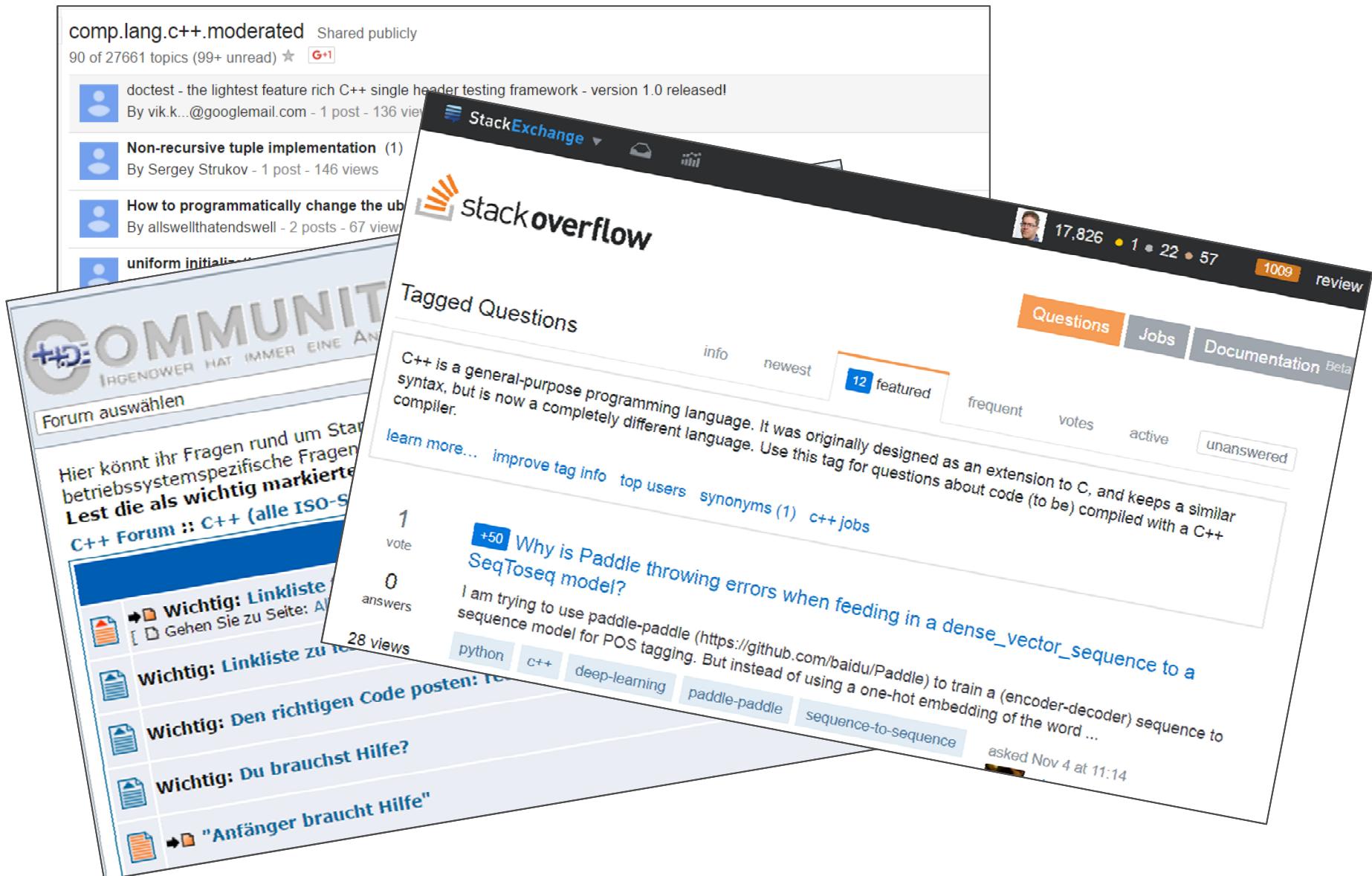
- Code has to be *extremely* well written to serve as a well documented resource for self-learning
- We can get lucky if we have a colleague or read code by someone who knows modern C++

They probably also know the quirks of our niche

- But even then
“knowledge inbreeding” – no fresh ideas
- And if not?

We learn the same outdated stuff the rest of the team is practicing

Forums, Usenet, Stackoverflow



Forums, Usenet, Stackoverflow

Pros:

- Multiple views, i.e. a community

Cons:

- Very specific detailed topics
- Too much noise
- Trolls

Conferences

Including conference videos



Conferences

Including conference videos

Pros:

- Videos are often freely available

Cons:

- Again only single bits and pieces
- Often relatively advanced topics
- Poor visibility

Outro

Wishes

What do we need for better learning resources?

And *teaching* resources, of course.

- Authority & visibility – to attract more students

When someone asks for a resource, there should be one or two answers, not tens or hundreds

- Flexibility or even volatility
 - To avoid outdated resources
 - To adapt resources to different needs (niches)
- A community
 - Healthy discussion about modern C++
 - More than a single opinion
 - Keep track with the evolution of C++

What we have

<https://isocpp.org/>

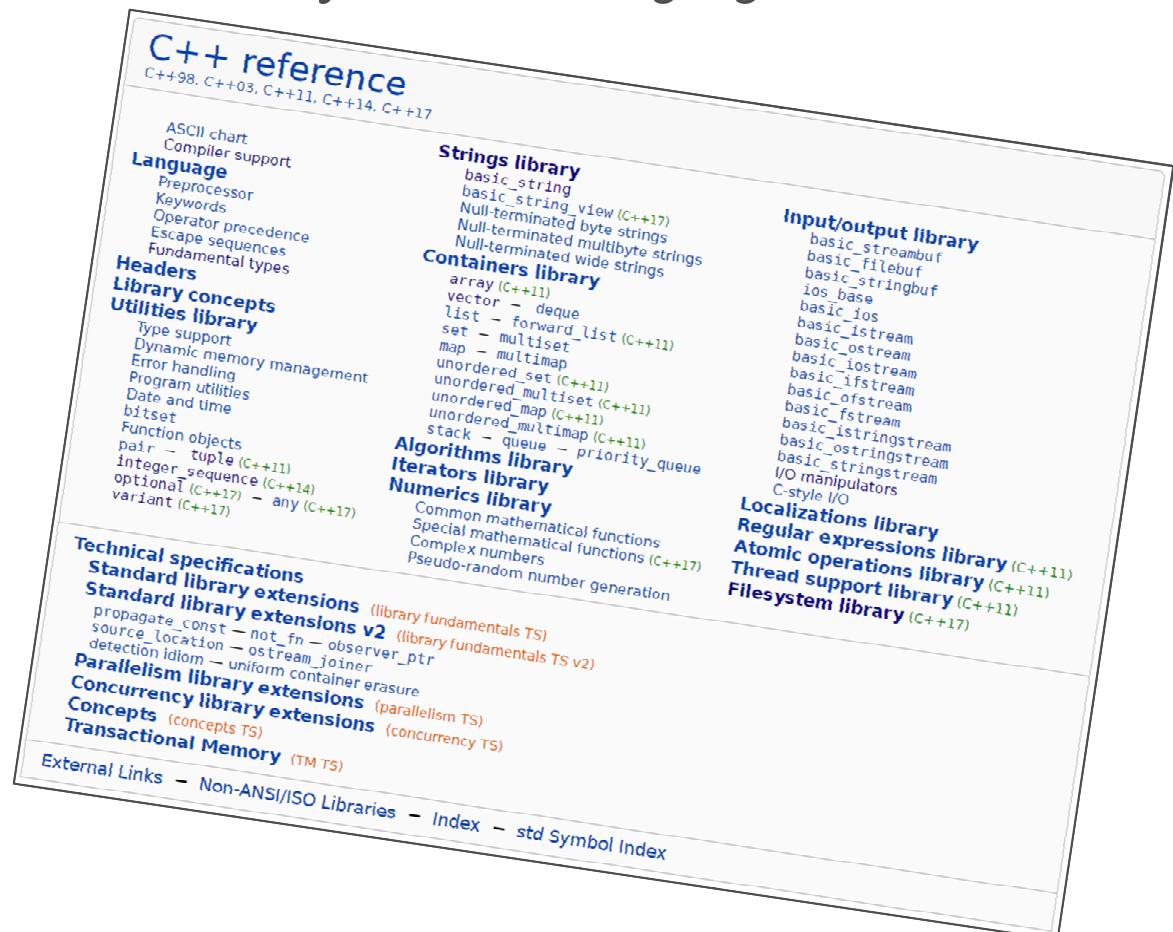
- Hub for C++-related things
- Forum
- Blog
 - Blogs & Books
 - Videos
 - Events & Training
- Standardization Info
 - Status, Notes,...
- C++-FAQ



What we have

cppreference.com

- Online reference for standard library and core language features
 - Wiki format
 - Up to date
- Still only a reference
 - Only few basics
 - No reading order
- Executable code



What we have

C++ Core Guidelines

Best practice guidelines accompanied by static analysis rules.

- Static analyzers can find a lot of code smells
- Core guidelines have rationale and alternatives for unsafe and outdated idioms
- <https://github.com/isocpp/CppCoreGuidelines>

What we have

StackOverflow Documentation

- A place for code examples and documentation
 - Like a collection of community-edited blog articles
- Still in beta
 - Not much content yet (currently 137 topics)
 - No apparent order

<https://stackoverflow.com/documentation/c%2b%2b/>

What I'd like to have...

... just fantasizing...

A Wiki-Blog similar to SO Documentation

- Where people can compile “books” by aggregating single pages
Different books for different needs
- Well-moderated discussions

Questions? Ideas? Comments?

Let's talk!

zühlke
empowering ideas

Thank you!



arne.mertz@zuehlke.com

Simplify C++!
ARNE-MERTZ.DE

cpplang.slack.com
cpplang.diegostamigni.com