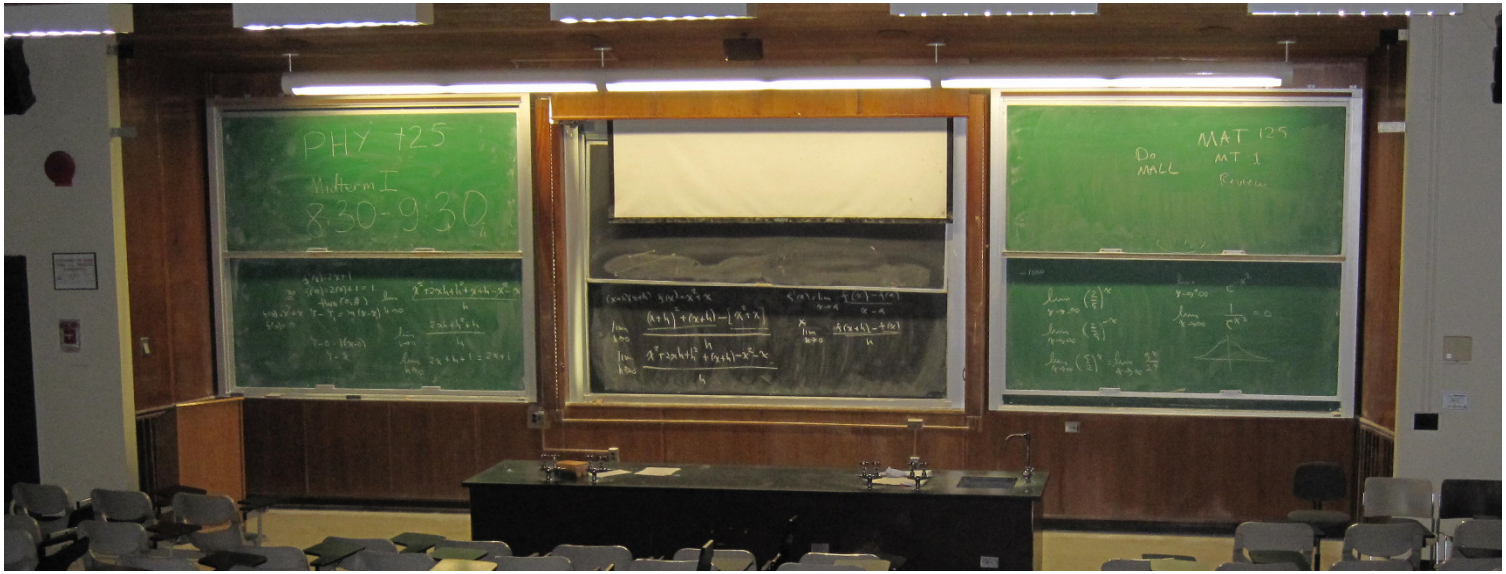
zühlke
empowering ideas



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

@arne_mertz

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 >= C++11

21% ask Alexandrescu

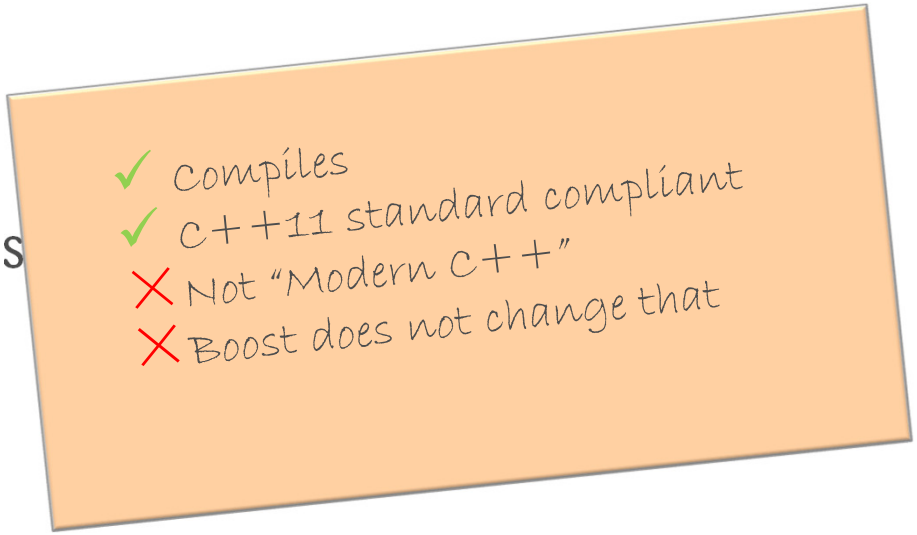
14% using C++ to its fullest

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



Meeting C++

@meetingcpp

So what is "Modern C++"?

~~13%~~ new standards + boost

~~52%~~ new standards >= C++11

21% ask Alexandrescu

14% using C++ to its fullest

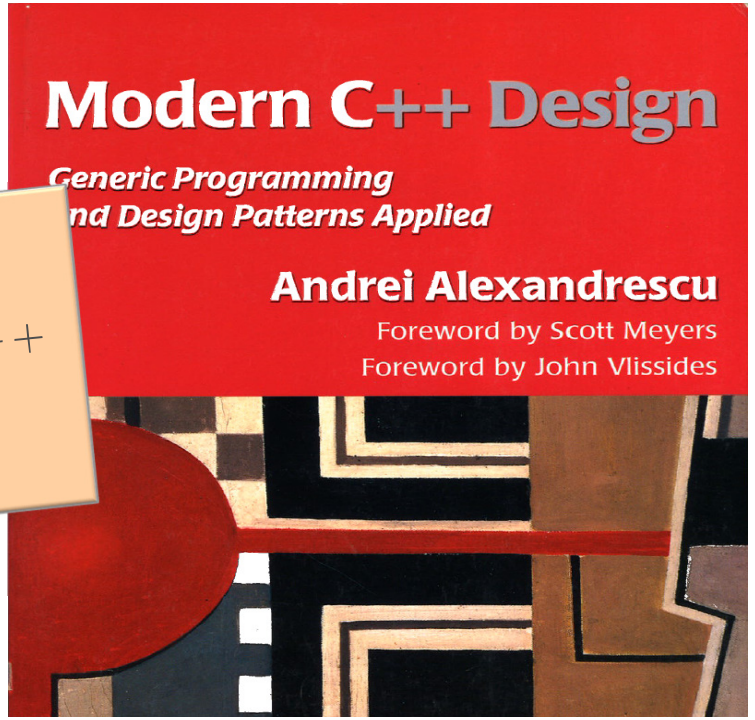
375 votes • Final results

“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

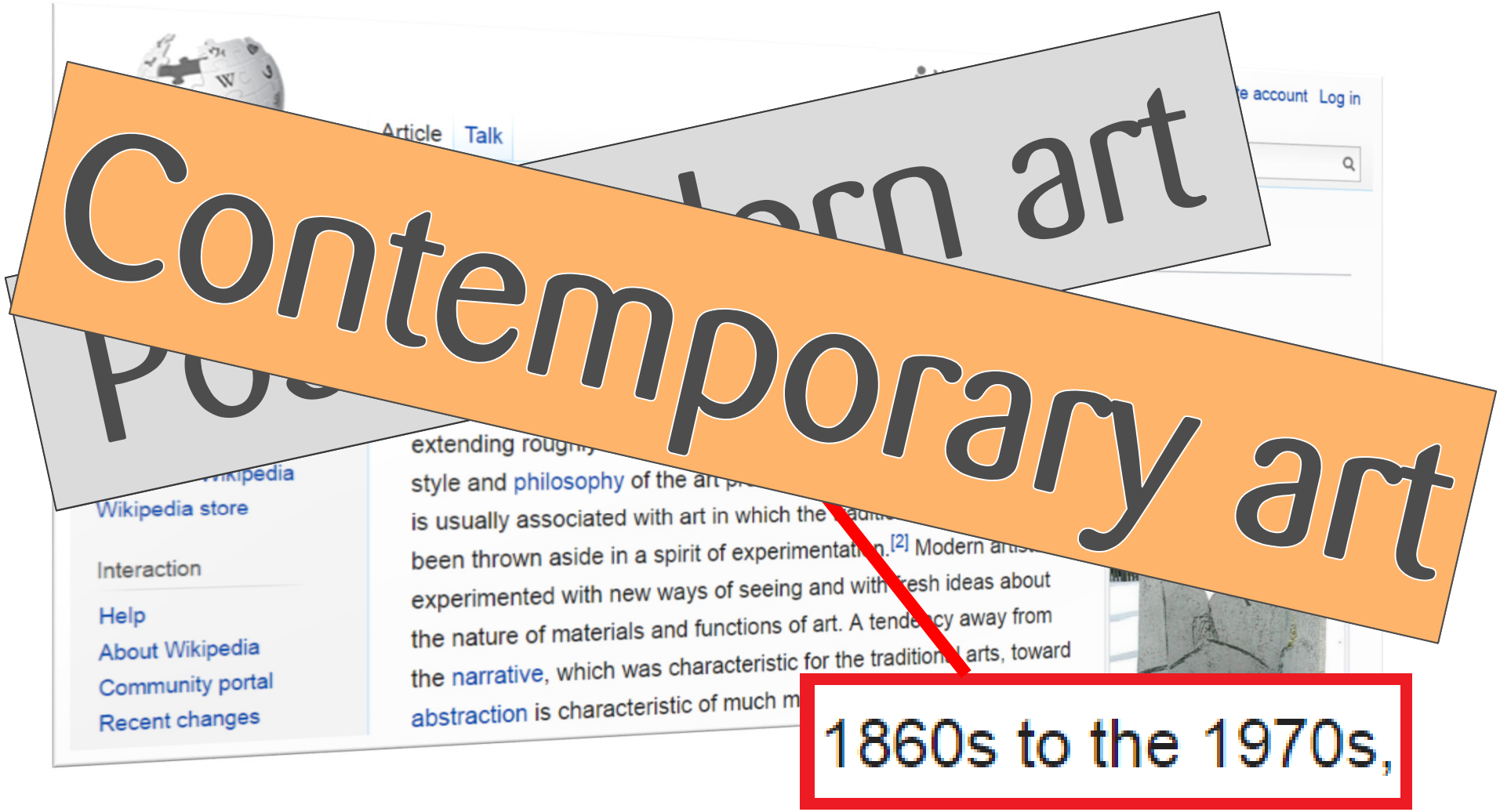
- Still a good read
- Influenced our thinking about C++
- But not “Modern C++” anymore



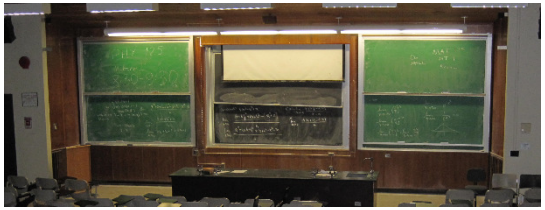
My understanding is that the book "Modern C++ Design" coined the term "modern C++". The term refers to a template-intensive, generic style of writing code.

Andrei Alexandrescu

Modern art




accu
2017
Functional C++ For Fun And Profit
Phil Nash
C++11 gave us lambd
boost::lambda) - so
more to functional
I'd even argue we
do functional pro
the language for the first time (if you ignore
nguage now, right? There's a bit
t class function objects.
mean we can't
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 interfa
libraries such as various improve
libraries, lo
Learning and Teaching Moder



zühlke
empowering ideas

Learning (and Teaching) the right mix of Modern,

Postmodern,

Functional,

Contemporary,

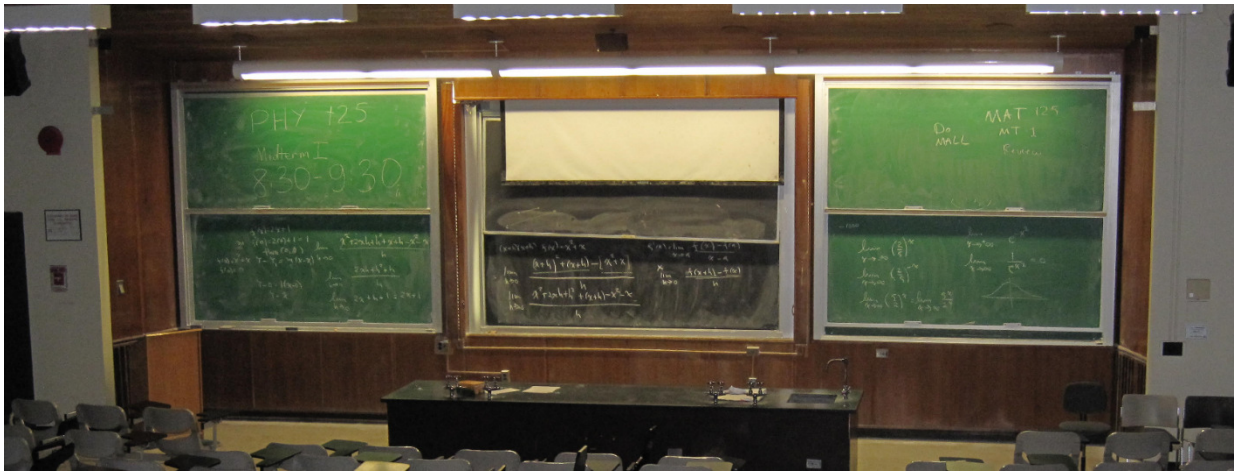
classical,

...

C++?

@arne_mertz

zühlke
empowering ideas



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

@arne_mertz

Modern

English
Oxford *Living* Dictionaries

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++"?

~~1%~~ new standards + boost

~~5%~~ new standards >= C++11

~~2%~~ ask Alexandrescu

14% using C++ to its fullest ✓

375 votes • Final results

Modern C++ is…

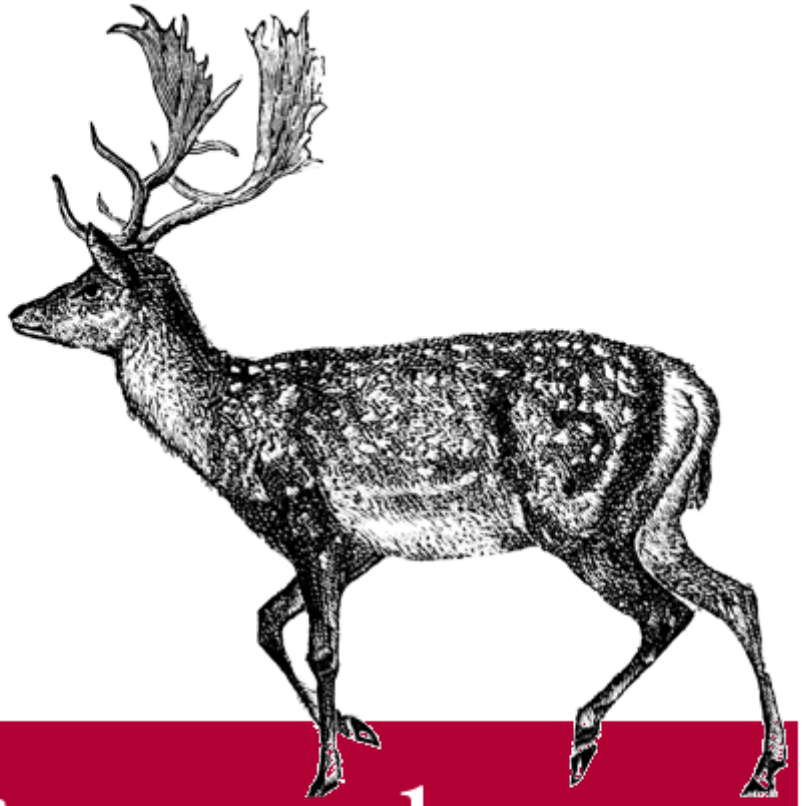
… what we consider best practice *today*

- 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

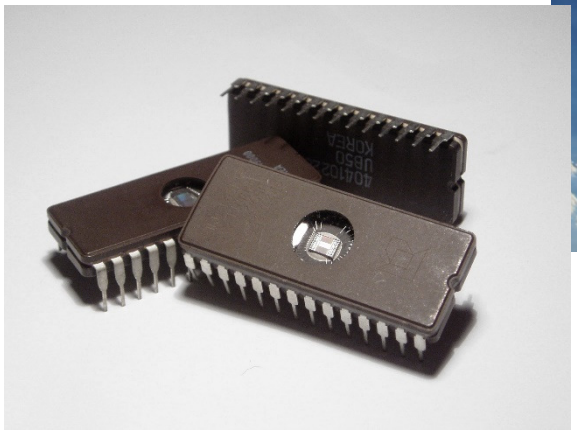
- Frequently new features
  - New standards currently every ~3 years
  - Plus TS
- But besides that, our *understanding* of the language continually changes
  - E.g. “universal references”
  - E.g. “Almost Always Auto”

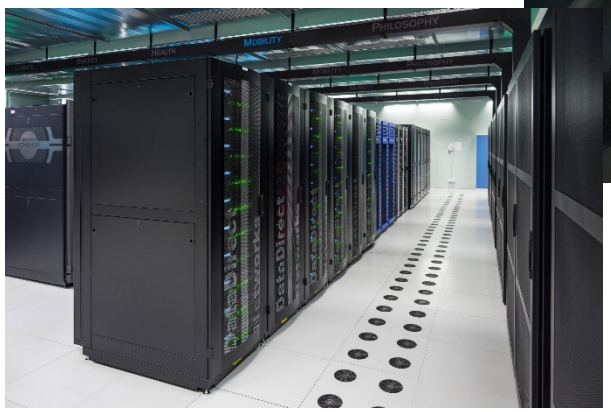
```
Widget w{"myWidget"};

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


Modern C++ is…

… a moving target

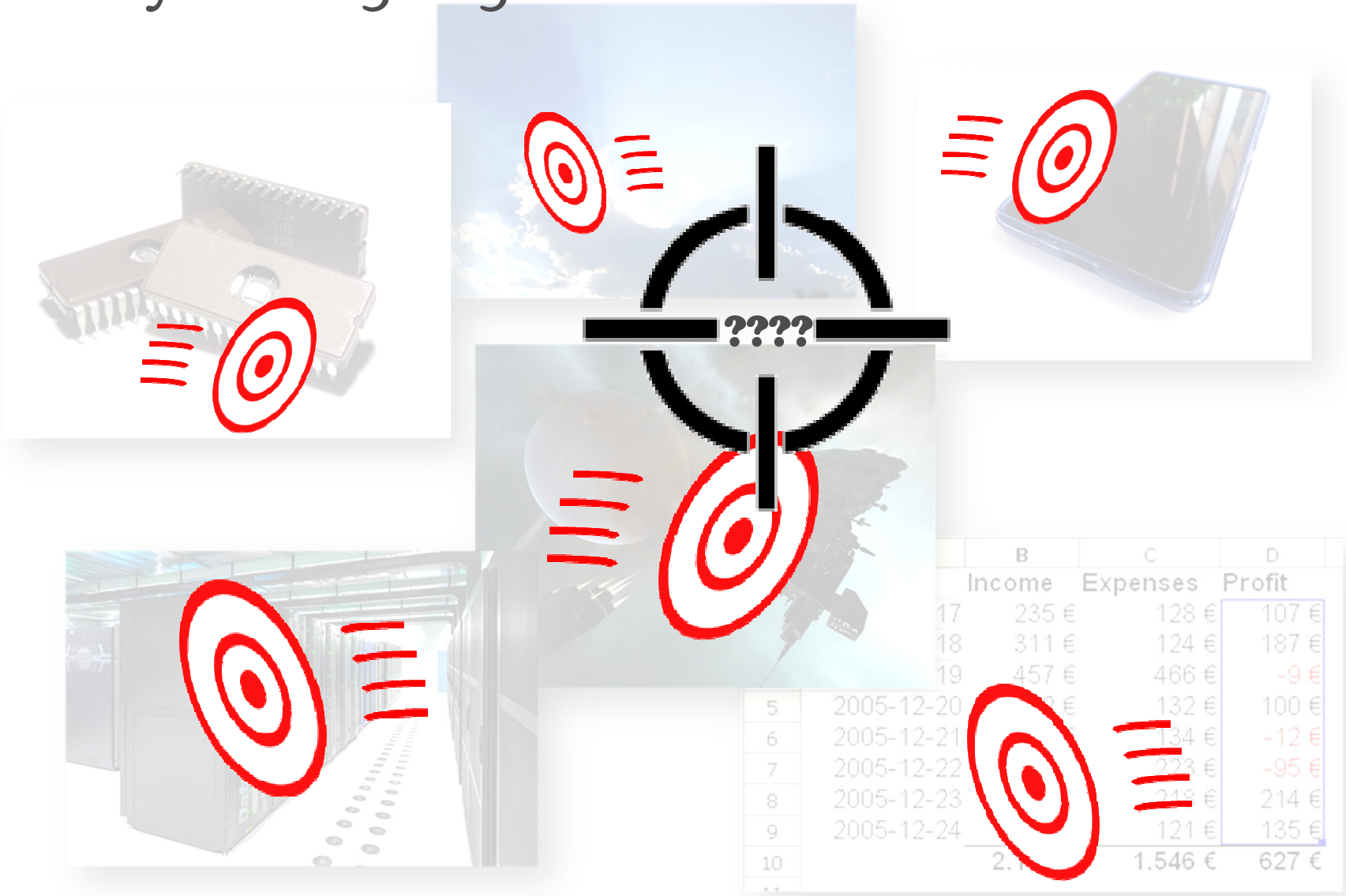
```
template <class K, class V>
void printMap(std::map<K, V> const& m) {
  for(std::map<K, V>::iterator it = m.begin(); it != m.end(); ++it) {
    K const& key = it->first;
    V const& value = it->second;
    std::cout << "(" << key << ": " << value << ")\n";
  }
}
template <class Map>
void printMap(Map 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 €	100 €
6	2005-12-21	122 €	134 €	-12 €
7	2005-12-22	128 €	223 €	-95 €
8	2005-12-23	432 €	218 €	214 €
9	2005-12-24	256 €	121 €	135 €
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

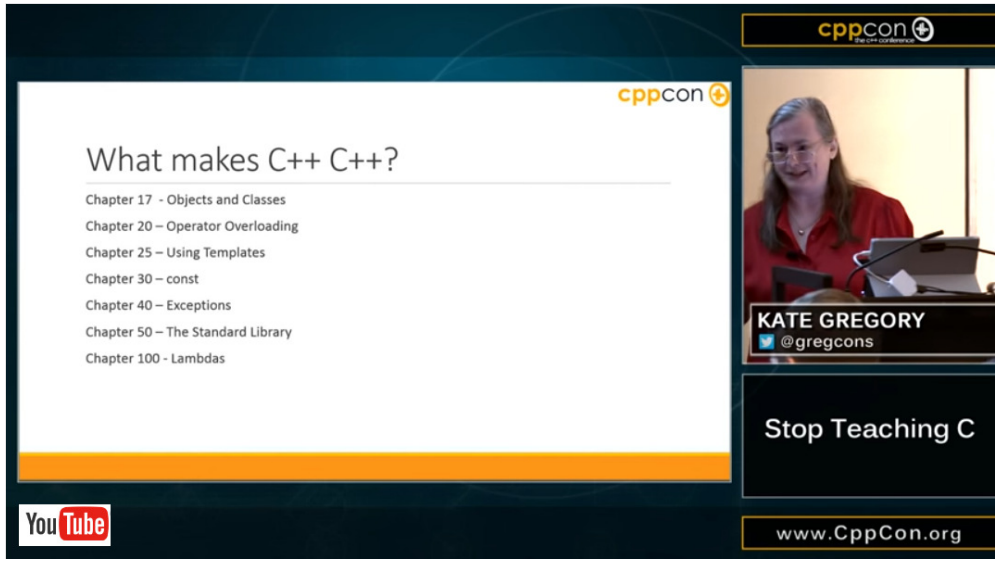
… for teachers

- What we see and learn first sticks best

→ Teach straight to the point instead of retelling our personal history

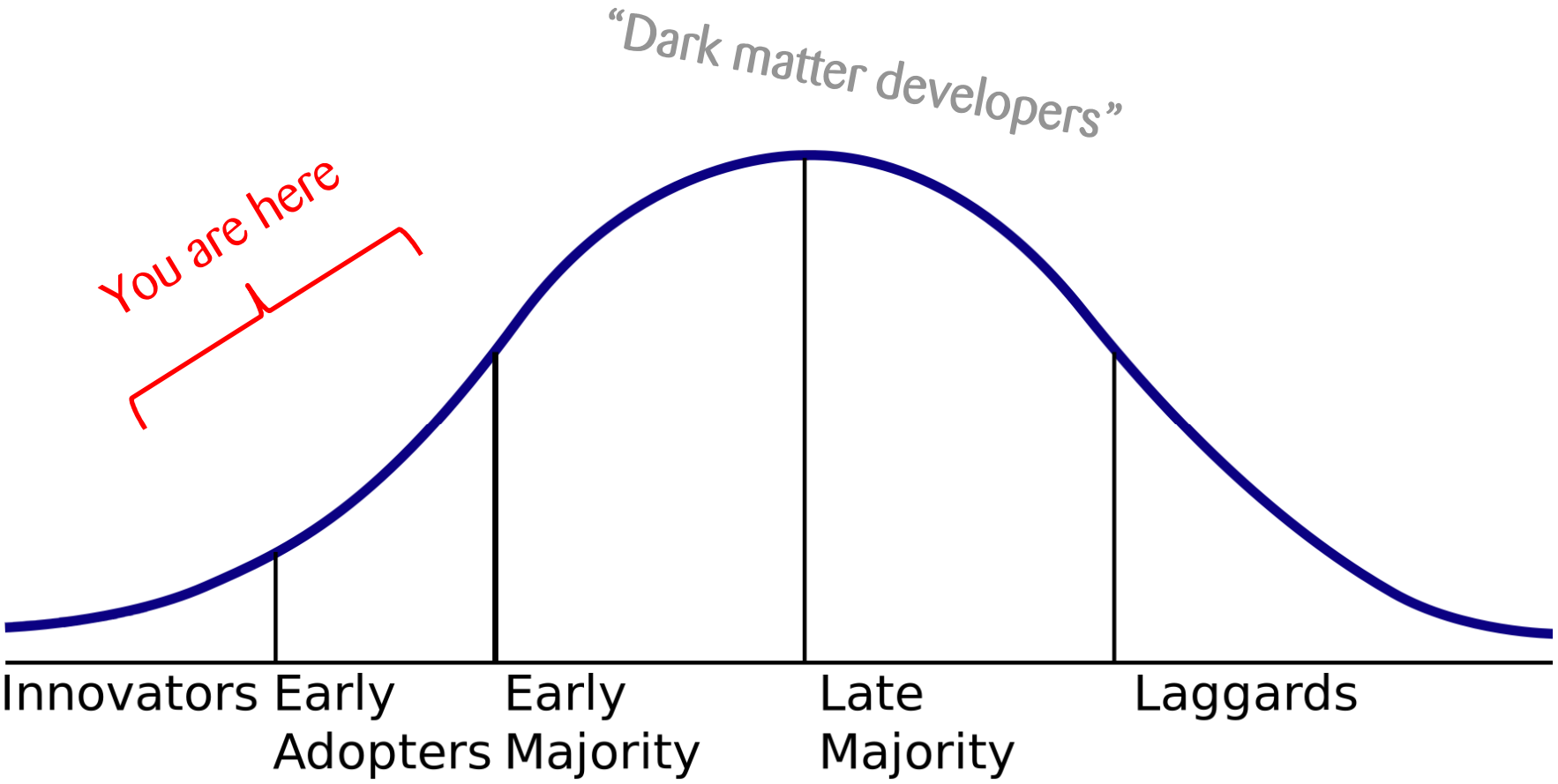
Example: start with make_unique & unique_ptr instead of new/delete → smart pointers

→ 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?

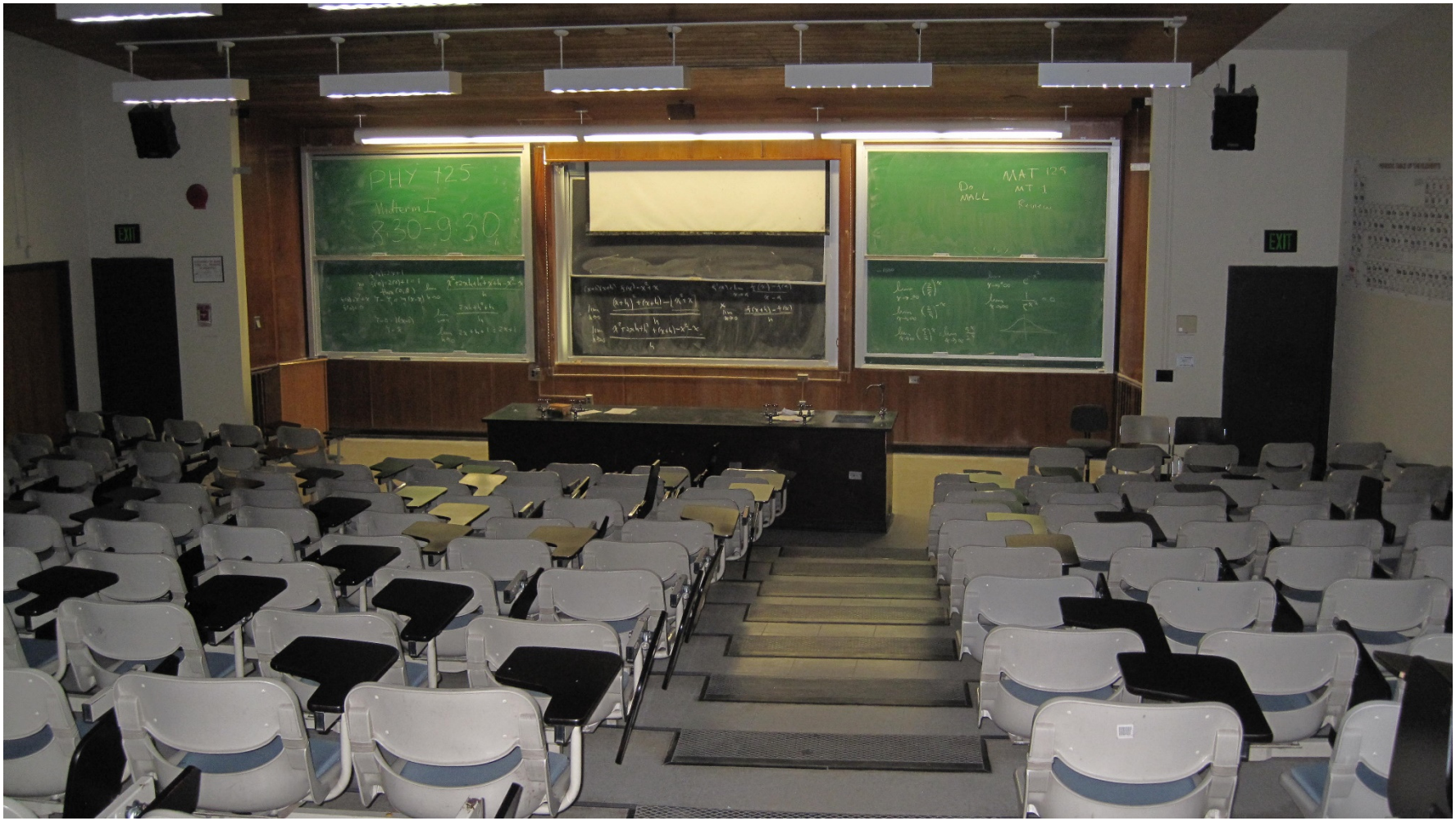
- **6%** Teacher/Professor
- **56%** Books
- **31%** Online Tutorials & Blogs
- **7%** Other (please specify)

469 votes • Final results

Other:
+ Colleagues
+ Reading code
+ Magazines
+ Usenet
+ Books!
+ Conference videos

Class room…

… and teachers/professors



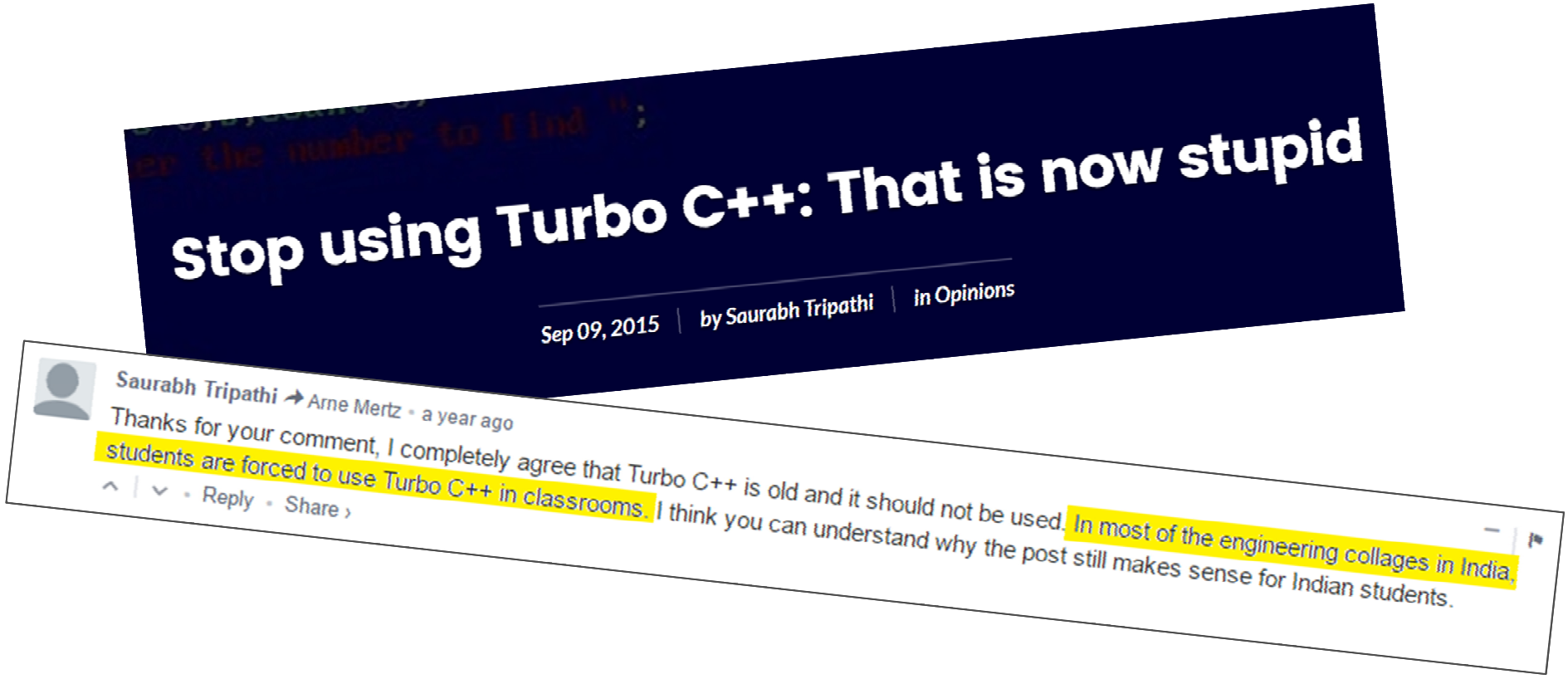
Class room…

… and teachers/professors

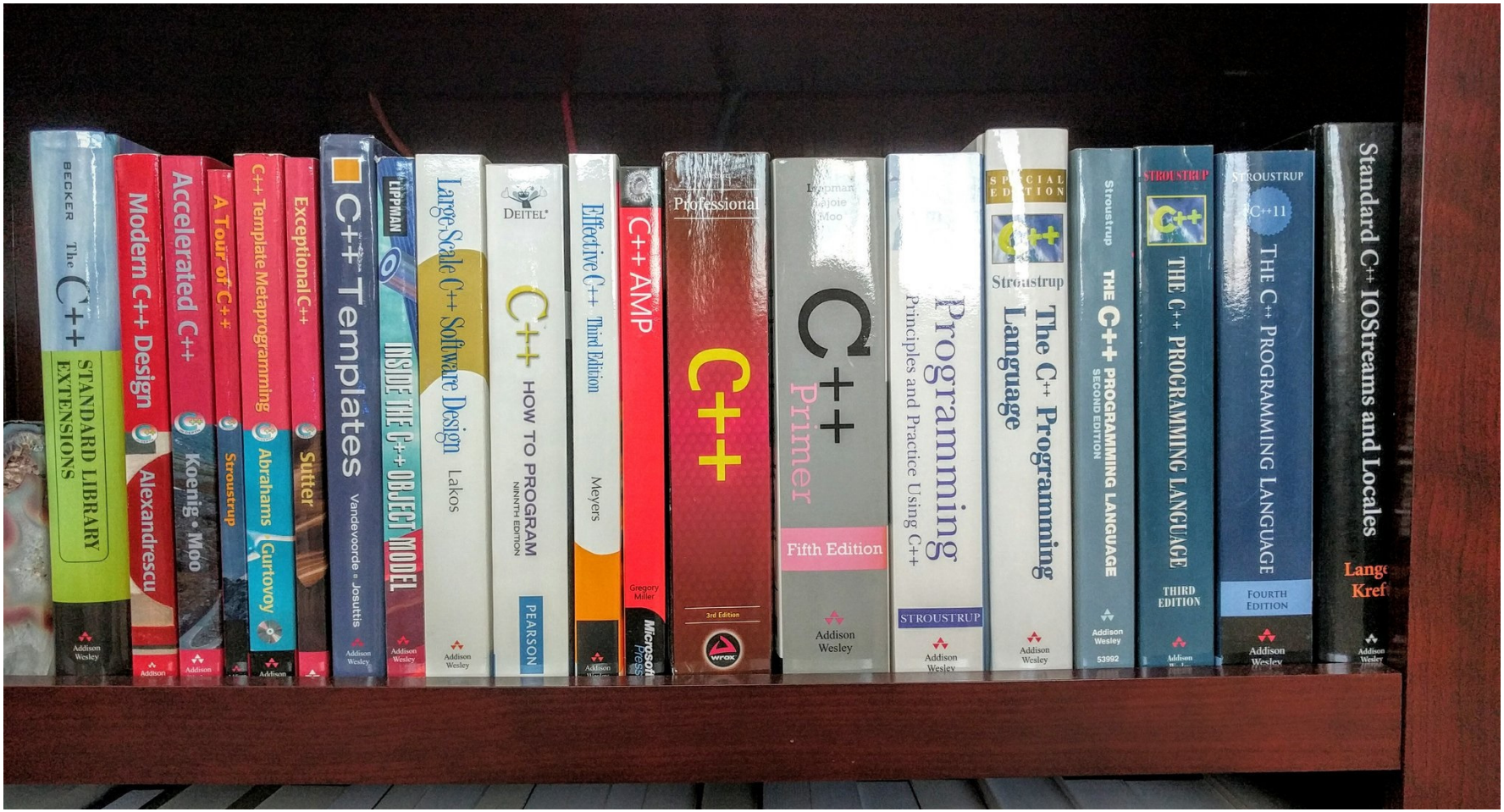
- Some teachers are enthusiastic and love to teach
- Others don't but *have* to
  - Prepare class once
  - Repeat for N years

  → fresh legacy C++ students

Class room…



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

0

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!".

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

Blogs & Tutorials

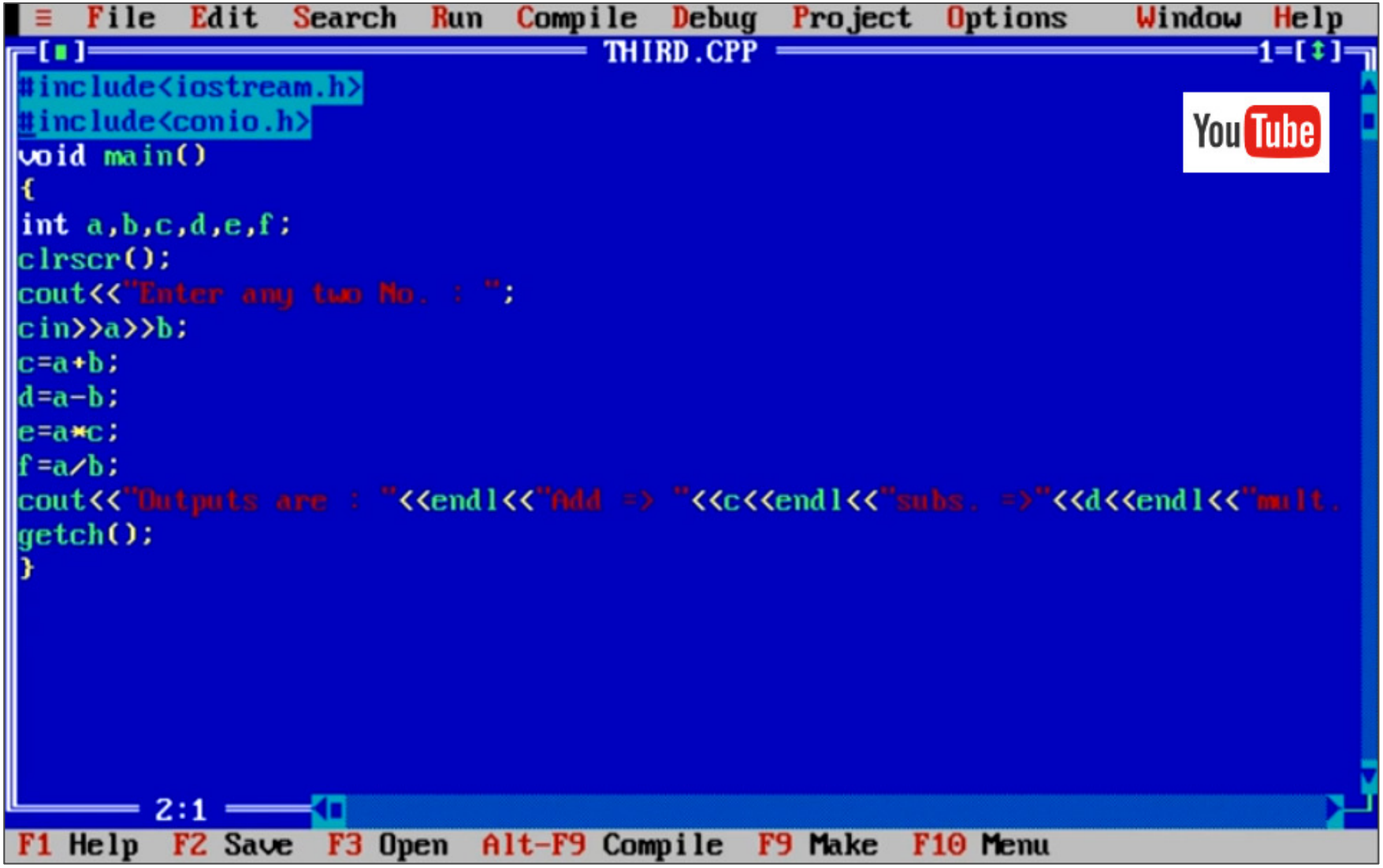
Pros:

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

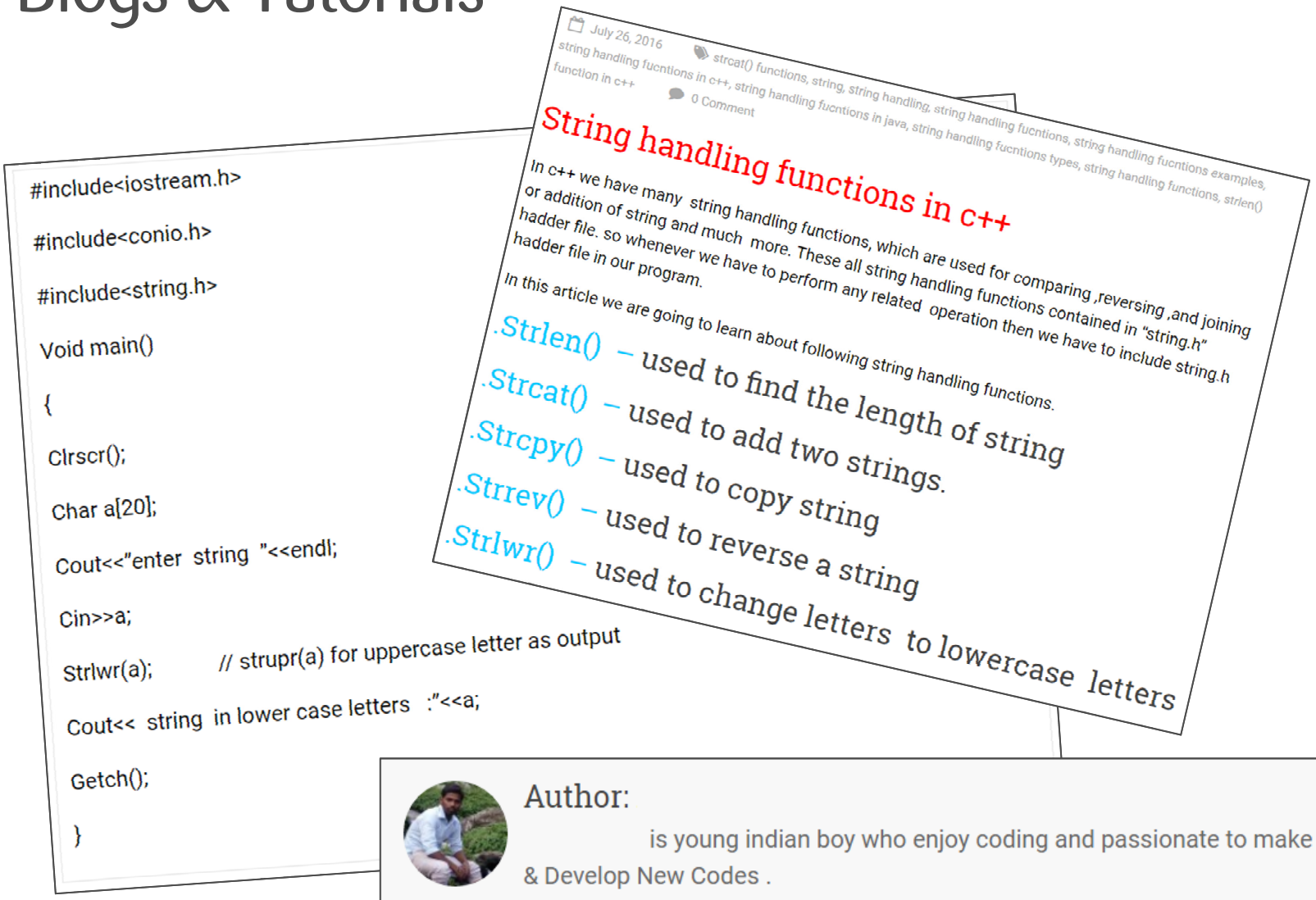
Cons:

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

Blogs & Tutorials



Blogs & Tutorials



Colleagues & reading Code

Colleagues & reading Code

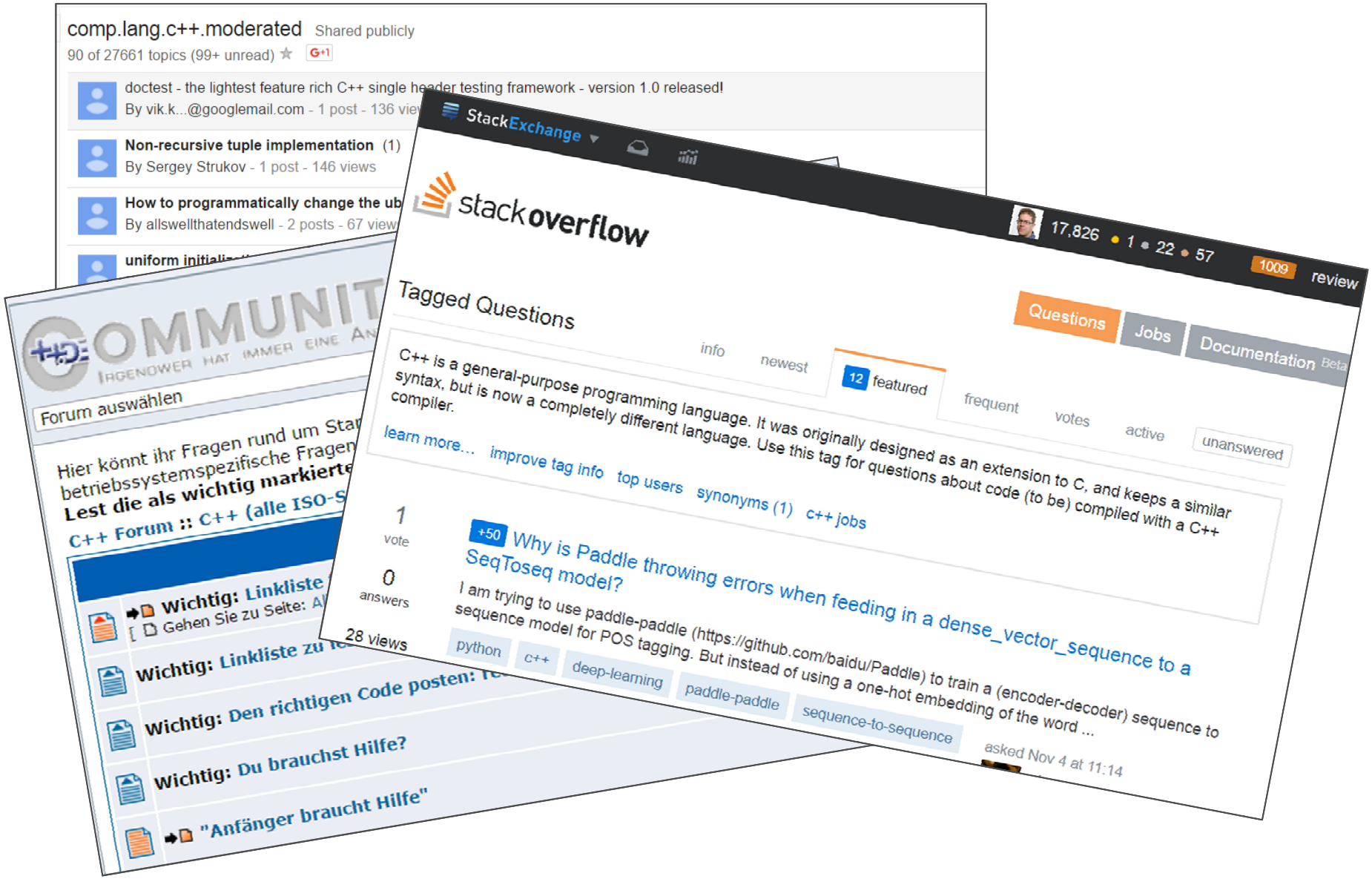
- Code has to be *extremely* well written to serve as a well documented resource for self-learning
- We can get lucky if we have a colleague or read code by someone who knows modern C++

  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

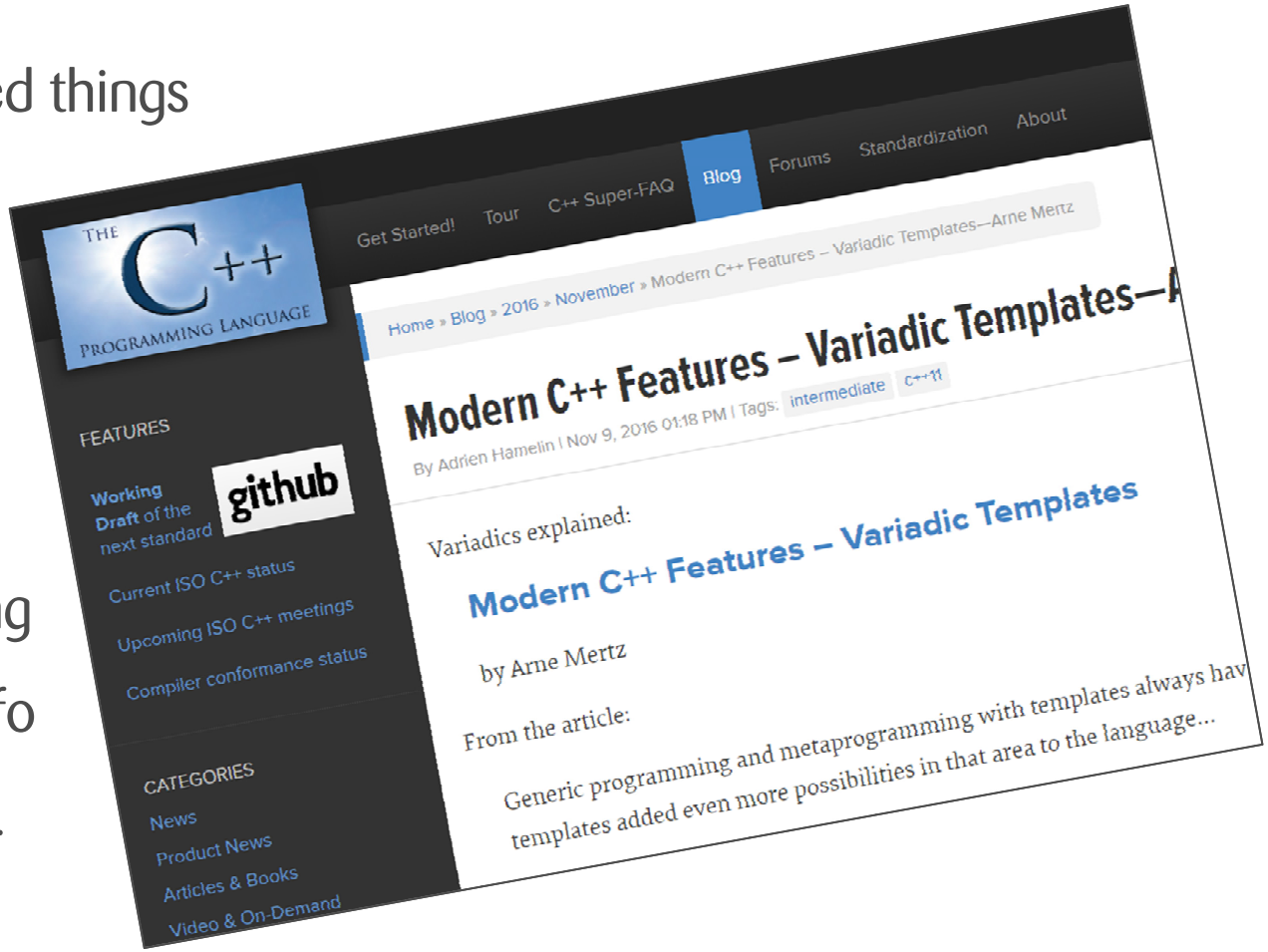
  When someone asks for a resource, there should be one or two answers, not tens or hundreds

- 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/

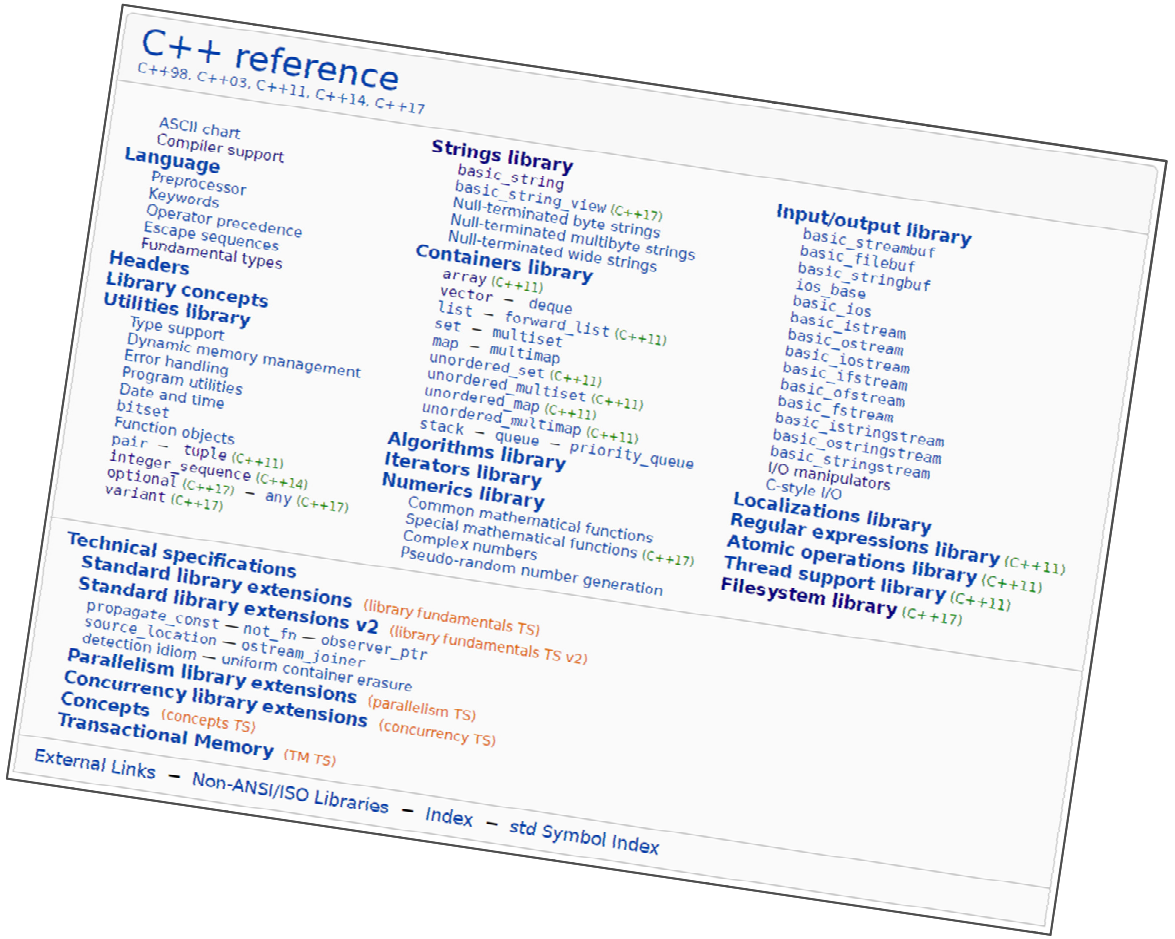
- Hub for C++-related things
- Forum
- Blog
  - Blogs & Books
  - Videos
  - Events & Training
- Standardization Info
  - Status, Notes,…
- C++-FAQ



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!

Thank you!


@arne_mertz

arne.mertz@zuehlke.com

Simplify C++!

ARNE-MERTZ.DE


cpplang.slack.com

cpplang.diegostamigni.com