0.12 — Configuring your compiler: Choosing a language standard

BY ALEX ON NOVEMBER 12TH, 2019 | LAST MODIFIED BY ALEX ON JANUARY 26TH, 2020

With many different versions of C++ available (C++98, C++03, C++11, C++14, C++17, etc...) how does your compiler know which one to use? Generally, a compiler will pick a standard to default to (often *not* the most recent language standard). If you wish to use a different standard, you'll have to configure your IDE/compiler to do so. These settings are applied to the active project only. You need to set them again when you create a new project.

Learncpp uses C++17. If your compiler isn't capable of C++17, you won't be able to follow all lessons, though the majority is unaffected. Make sure you're using an up-to-date compiler so you can compile all the examples yourself. There's an example at the end of this lesson which you can use to test if you set up your compiler correctly.

Code names for in-progress language standards

Note that the language standards are named after the years in which they are finalized (e.g. C++17 was finalized in 2017).

However, when a new language standard is being agreed upon, it's not clear in what year the finalization will take place. Consequently, in-progress language standards are given code names, which are then replaced by the actual names upon finalization of the standard. For example, *C++11* was called *c++1x* while it was being worked on. You may still see the code names used in places (especially for upcoming version of the language standard, which won't have a final name yet).

Here's a mapping of code names to the final names:

- c++1x = C++11
- c++1y = C++14
- c++1z = C++17
- c++2a = C++20

For example, if you see `c++1z`, this is synonymous with the C++17 language standard.

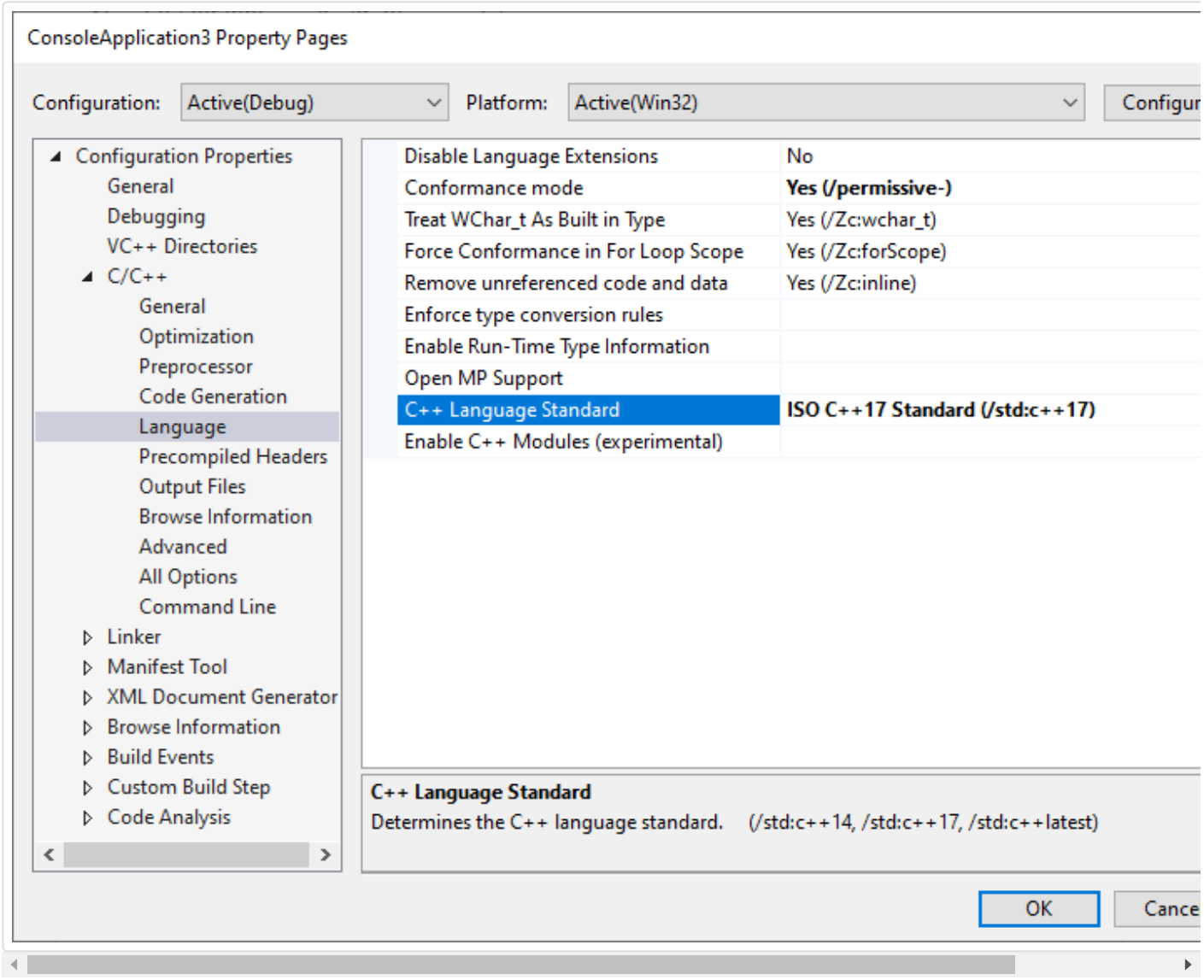
Setting a language standard in Visual Studio

As of the time of writing, Visual Studio 2019 defaults to C++14 capabilities, which does not allow for the use of newer features introduced in C++17 (and C++20), some of which are covered in future lessons.

To use these newer features, you'll need to enable a newer language standard. Unfortunately, there is currently no way to do this globally -- you must do so on a project-by-project basis.

To do so, open your project, then go to *Project menu* > *(Your application's Name)* Properties, then open *Configuration Properties* > *C/C++* > *Language*.

From there, you can set the *C++ Language Standard* to the version of C++ you wish to use.



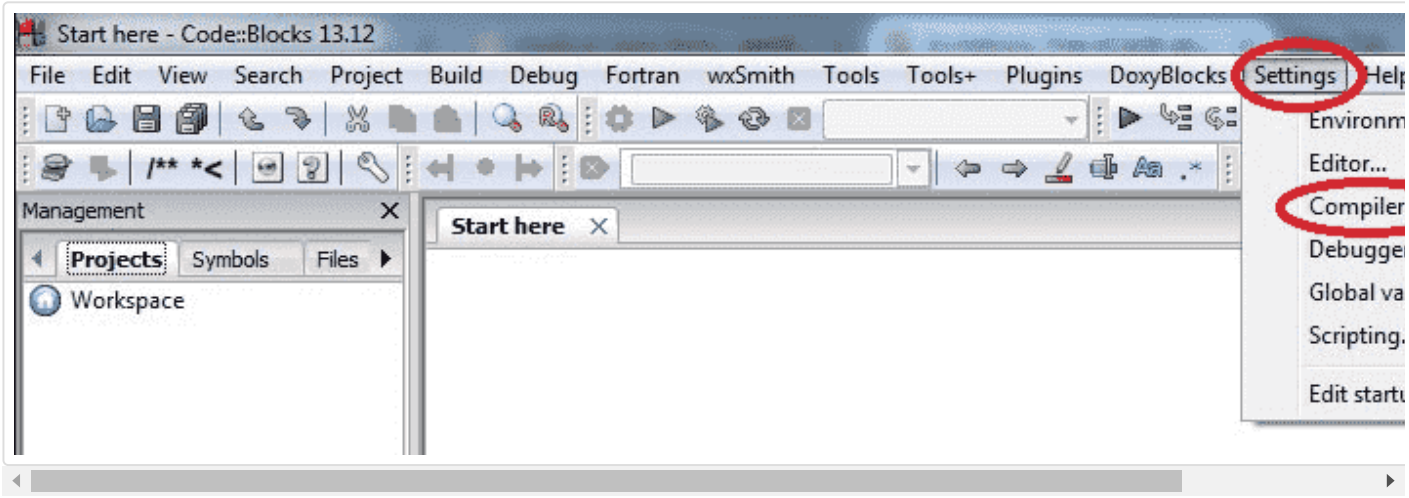
As of the time of writing, we recommend selecting “ISO C++17 Standard (/std:c++17)”, which is the latest stable standard.

If you want to experiment with capabilities from the upcoming C++20 language standard, choose “ISO C++ Latest (/std:c++latest)” instead. Just note that support may be spotty or buggy.

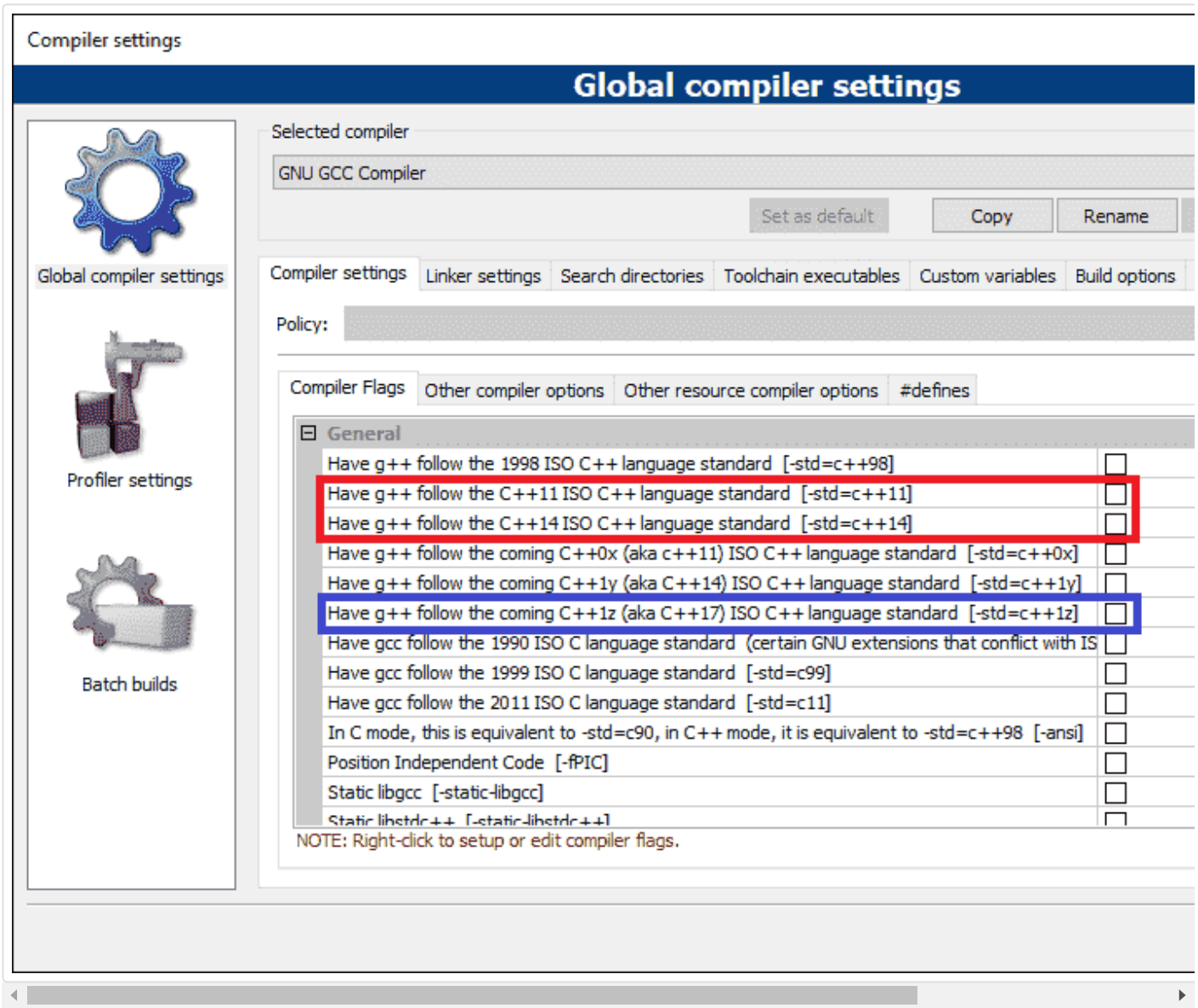
Setting a language standard in Code::Blocks

Code::Blocks may default to a pre-C++11 language standard. You’ll definitely want to check and ensure a more modern language standard is enabled.

The good news is that Code::Blocks allows setting your language standard globally, so you can set it once (rather than per-project). To do so, go to *Settings menu* > *Compiler*:



Then find the box or boxes labeled *Have g++ follow the C++XX ISO C++ language standard [-std=c++XX]*, where XX is 11, 14, or some other higher number (see the items inside the red box below for examples):



Check the one with the highest number (in the above image, that's the C++14 option inside the red box).

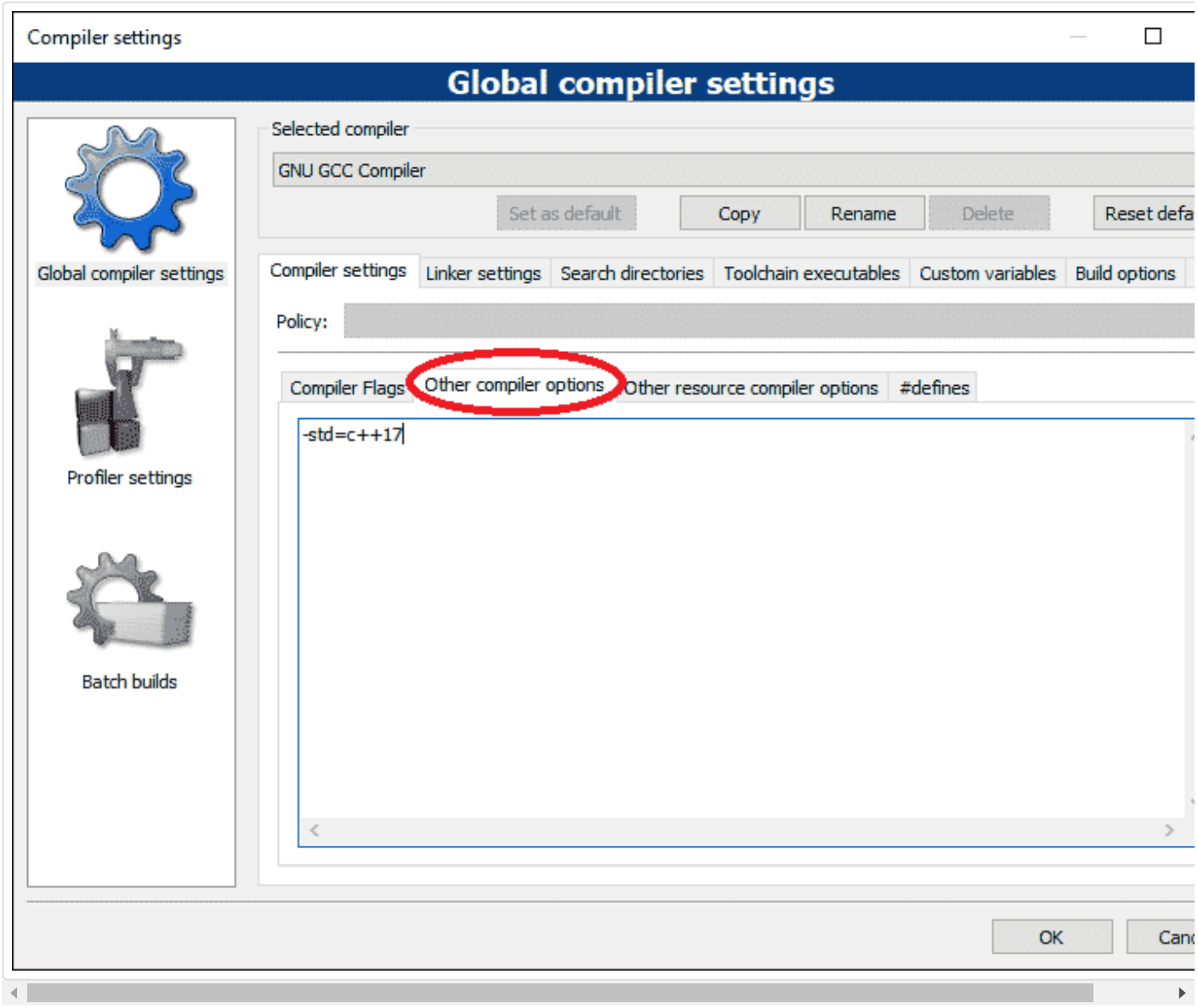
Your version of Code::Blocks may also have support for upcoming (or just released) versions of C++. If so, these will be labeled *Have g++ follow the coming C++XX (aka C++YY) ISO C++ language standard [-std=c++XX]* (see the blue box

above). You can optionally check these if you would like to enable features added in that version, but note that support may be incomplete (ie. some features may be missing).

See the list at the top of the lesson for a list of codenames to language standards mappings.

For example, if you want to enable C++17 capabilities and your settings doesn't have a C++17 option, look for `-std=c++1z` (the code name for C++17).

Alternatively, you can go to the Other Compiler Options tab and type in *-std=c++17*.



Note

This will work if your compiler has C++17 support. If you're using an older version of Code::Blocks and C++17 features don't seem to work, upgrade your compiler.

Setting a language standard in g++

For GCC/G++ users

For GCC/G++, you can pass compiler flags -std=c++11, -std=c++14, -std=c++17, or -std=c++2a to enable C++11/14/17/2a support respectively.

Testing your compiler

After you enabled C++17 or higher, you should be able to compile the following code without any warnings or errors.

```
#include <array>
#include <iostream>
#include <string_view>
#include <tuple>
#include <type_traits>

namespace a::b::c
{
    inline constexpr std::string_view str{ "hello" };
}

template <class... T>
std::tuple<std::size_t, std::common_type_t<T...>> sum(T... args)
{
    return { sizeof...(T), (args + ...) };
}

int main()
{
    auto [iNumbers, iSum]{ sum(1, 2, 3) };
    std::cout << a::b::c::str << ' ' << iNumbers << ' ' << iSum << '\n';

    std::array arr{ 1, 2, 3 };

    std::cout << std::size(arr) << '\n';

    return 0;
}
```

If you can't compile this code, you either didn't enable C++17, or your compiler doesn't fully support C++17. In the latter case, please install the latest version of your IDE/compiler, as described in lesson **0.6 -- Installing an Integrated Development Environment (IDE)**.

1.1 -- Statements and the structure of a program

Index

0.11 -- Configuring your compiler: Warning and error levels

C++ TUTORIAL | PRINT THIS POST

24 comments to 0.12 — Configuring your compiler: Choosing a language standard

PAPA
February 6, 2020 at 2:10 pm · Reply

Amazing course!

Aditya
January 25, 2020 at 6:19 am · Reply

I am using Code::Blocks, the 17.12 version and I got this error even after doing everything that is part of this tutorial,

|3|fatal error: string_view: No such file or directory|

Could it have anything to do with "..features added in that version, but note that support may be incomplete (e.g. some features may be missing)."
What do I do??

And also,
Is it okay if I have zero knowledge of C++ before beginning these tutorials?

nascardriver
January 25, 2020 at 7:18 am · Reply

The compiler you're using with code::blocks probably doesn't support C++17. There are currently no instructions for manually adding a compiler to code::blocks here.
Googling for "code blocks C++17 windows" (Replace windows with your OS) should get you good instruction on how to add another compiler.

Don't quit if you can't get it to work, you can still follow the majority of the tutorials with an old compiler.

The tutorials are targeted at beginners, you don't need any prior programming knowledge.

Aditya
January 26, 2020 at 6:35 am · Reply

Oh okay, thank you!

Alex
January 28, 2020 at 11:48 am · Reply

I added some instructions to lesson 0.6 on how to upgrade your Code::blocks compiler to a newer C++17 compatible version.

Sam
January 30, 2020 at 1:01 pm · Reply

Thank you for continuing to update these guides in our post-Blade runner world. You're a star, Alex.

Vitaliy Sh.

December 26, 2019 at 6:43 am · Reply

Typo: two "[/note]"s is being visible near the end of the lesson.

Vitaliy Sh.
December 25, 2019 at 6:33 pm · Reply

```
1 <img src="https://www.learncpp.com/images/CppTutorial/Chapter0/CB-C++11-min.png" alt="Code::Blocks C++11 setting">
```

Maybe your rather change "alt" to better fit the lesson's text: like a "CB: choose the stable/in-progress standard to follow" or smth?

Vitaliy Sh.
December 25, 2019 at 6:03 pm · Reply

Hi Sires!

```
1 <img src="https://www.learncpp.com/images/CppTutorial/Chapter0/VS2019-Project-Language-min.png" alt="">
```

Empty "alt" attribute here.

Vitaliy Sh.
November 17, 2019 at 8:24 am · Reply

...
(since it isn’t clear what year the standards will be finalized in until it actually happens).
...

Maybe a typos (hard sentence to me):
1) Maybe use "which" instead of "what"?
2) No comma after "finalized in".

Is my proposal from https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-choosing-a-language-standard/comment-page-1/#comment-437277 bad?

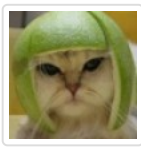
Alex
November 17, 2019 at 9:52 am · Reply

1) Generally "which" is correct when we're selecting from a limited choice, and "what" is used otherwise. What seems like the better choice here, since we aren't given a discrete set of options.
2) Fixed. Thanks!

Your other proposal isn't bad, it just didn't seem to be to be better than what was already written.

Vitaliy Sh.
November 18, 2019 at 3:43 am · Reply

``No comma after "finalized in"'' isn't fixed, somehow.



Alex
November 19, 2019 at 11:47 am · Reply

As far as I can tell, it shouldn't have one, per https://www.roanestate.edu/owl/Commas.html

Vitaliy Sh.
November 20, 2019 at 2:20 am · Reply

...
Note that language standards use code names prior to the standards being finalized (since it isn’t clear what year the standards will be finalized in until it actually happens).
...

Note that language standards use code names (instead of years) prior to the standards being finalized (since the year of finalization isn't clear until it actually happens).

?
PS: (If someone can't connect to www.roanestate.edu too: look up the https://www.lexico.com/en/grammar/punctuation or elsewhere).

Vitaliy Sh.
November 15, 2019 at 1:45 am · Reply

...
to enable C++11/14/17 support respectively.
...

Maybe add, just in case:
The last "-std=" wins.

OK:
g++ -std=c++03 -std=c++11 test.cpp
Err:
g++ -std=c++11 -std=c++03 test.cpp

Vitaliy Sh.
November 15, 2019 at 12:11 am · Reply

...
You can optionally check these if you would like to enable features in that version,
...

Typo: either "in" instead of "from", or no "added" after "features".

nascardriver
November 15, 2019 at 3:10 am · Reply

Thanks for all your feedback. Lesson updated.

Vitaliy Sh.
November 14, 2019 at 10:34 pm · Reply

...
C++ language standard [-std=c++XX]:, where XX is
...

Possible typo: ":" before comma.

Vitaliy Sh.
November 14, 2019 at 10:13 am · Reply

...
c++1x = C++11
c++1y = C++14
c++1z = C++17
c++2a = C++20
...

Possibly Typo: Capital "C" at right side from "=". Both g++ and clang++ are not accepted C++XX as a -std=.

Aaron Liu
December 9, 2019 at 10:47 pm · Reply

Definitly not a typo. This is a mapping from the codenames to the official names, which have the "C" capitalized.

Vitaliy Sh.
December 10, 2019 at 1:49 am · Reply

Thanks sir, i get it.

Vitaliy Sh.
November 14, 2019 at 10:06 am · Reply

...
Note that language standards use code names prior to the standards being finalized (since it isn’t clear what year the standards will be finalized until it actually happens).
...

If apropriate, to:

Note that prior to their finalization, the language standards are get referred by using the code names (since it isn't clear what year the standard will be finalized until it actually happens).

Vitaliy Sh.
November 14, 2019 at 9:29 am · Reply

...
Generally, a compiler will pick a standard to default to (often not the most recently language standard).
...

Typo: either no "published" between "recently" and "language", or "recently" like to be "recent".

Maybe also change to:
Generally, the compiler will pick a default one (which is often ...

Aymen Chehaider
November 13, 2019 at 1:45 pm · Reply

Nice tutorial Alex. Could you post some guidelines / steps on how to use the last compiler using MSYS2. Iknow that it may be out of scope or can find some clues by googling but will appreciate any stuff on how to configure MSYS2.

Thank you in advance