0.10 — Configuring your compiler: Compiler extensions

BY ALEX ON SEPTEMBER 19TH, 2018 | LAST MODIFIED BY NASCARDRIVER ON JANUARY 3RD, 2020

The C++ standard defines rules about how programs should behave in specific circumstances. And in most cases, compilers will follow these rules. However, many compilers implement their own changes to the language, often to enhance compatibility with other versions of the language (e.g. C99), or for historical reasons. These compiler-specific behaviors are called **compiler extensions**.

Writing a program that makes use of a compiler extension allows you to write programs that are incompatible with the C++ standard. Programs using non-standard extensions generally will not compile on other compilers (that don't support those same extensions), or if they do, they may not run correctly.

Frustratingly, compiler extensions are often enabled by default. This is particularly damaging for new learners, who may think some behavior that works is part of official C++ standard, when in fact their compiler is simply over-permissive.

Because compiler extensions are never necessary, and cause your programs to be non-compliant with C++ standards, we recommend turning compiler extensions off.

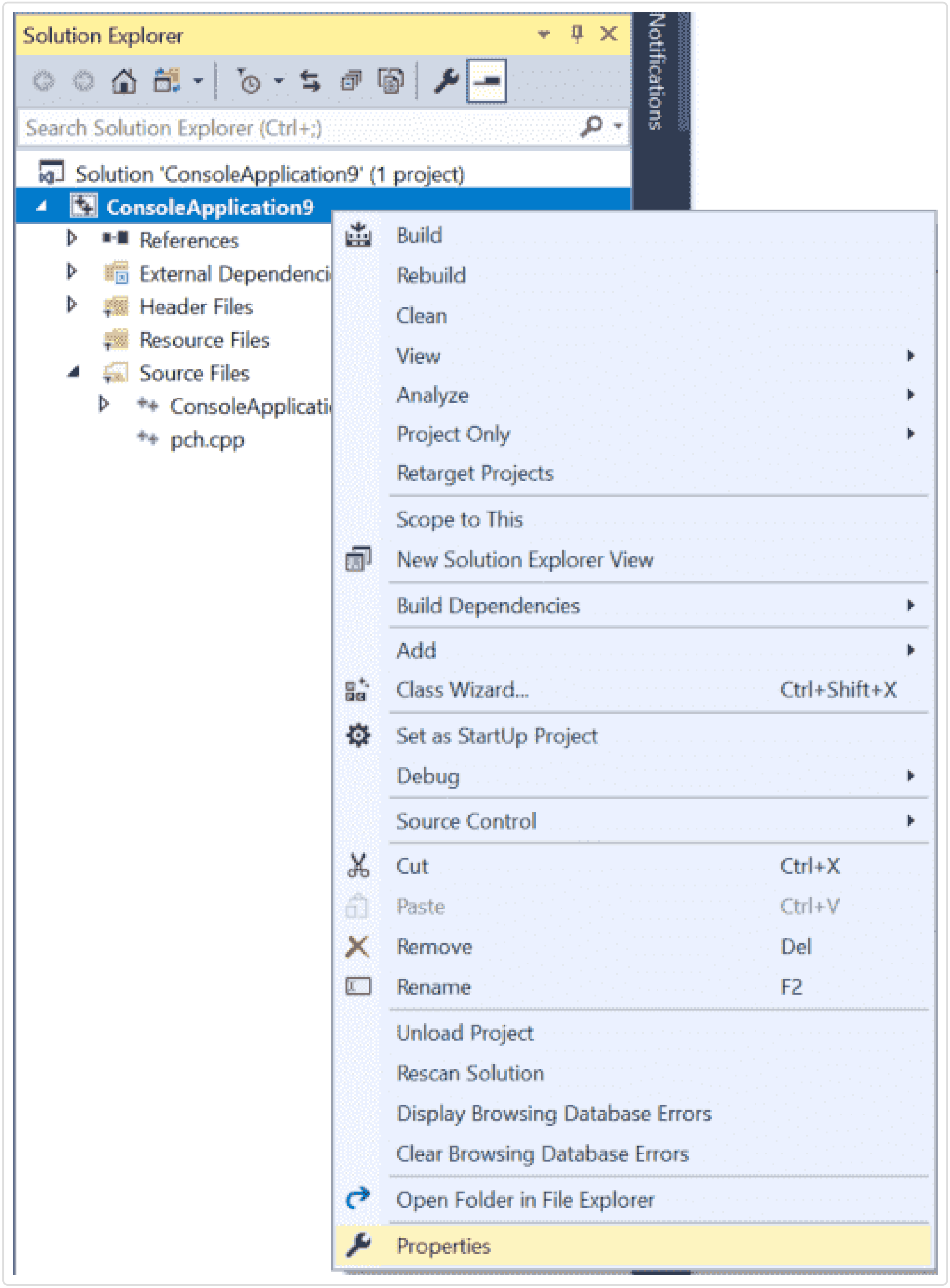
Best practice

Disable compiler extensions to ensure your programs (and coding practices) remain compliant with C++ standards and will work on any system.

Disabling compiler extensions

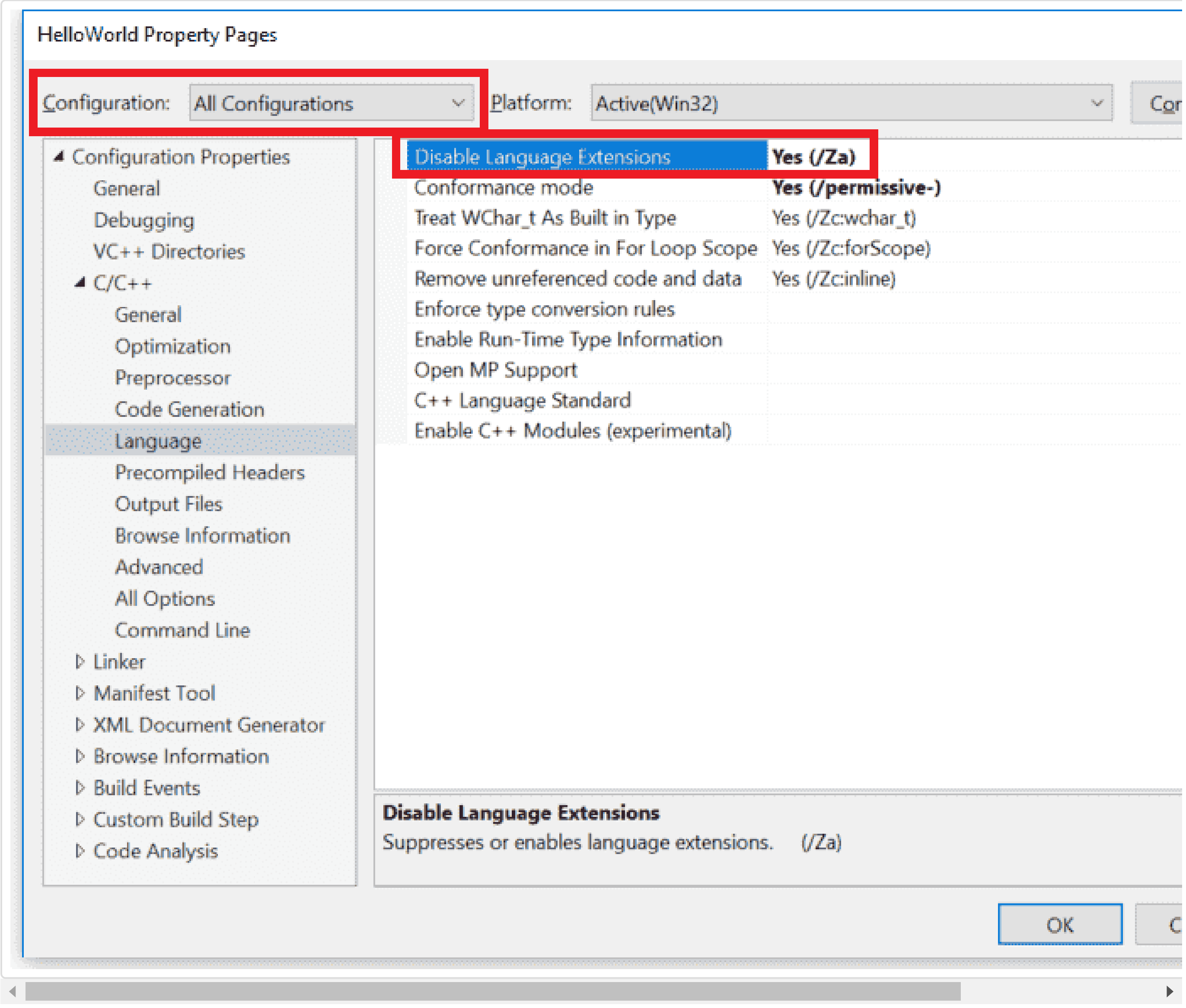
For Visual Studio users

To disable compiler extensions, right click on your project name in the *Solution Explorer* window, then choose *Properties*:



From the *Project* dialog, first make sure the *Configuration* field is set to *All Configurations*.

Then, click *C/C++ > Language tab*, and set *Disable Language Extensions* to *Yes (/Za)*.


HelloWorld Property Pages

Configuration: All Configurations

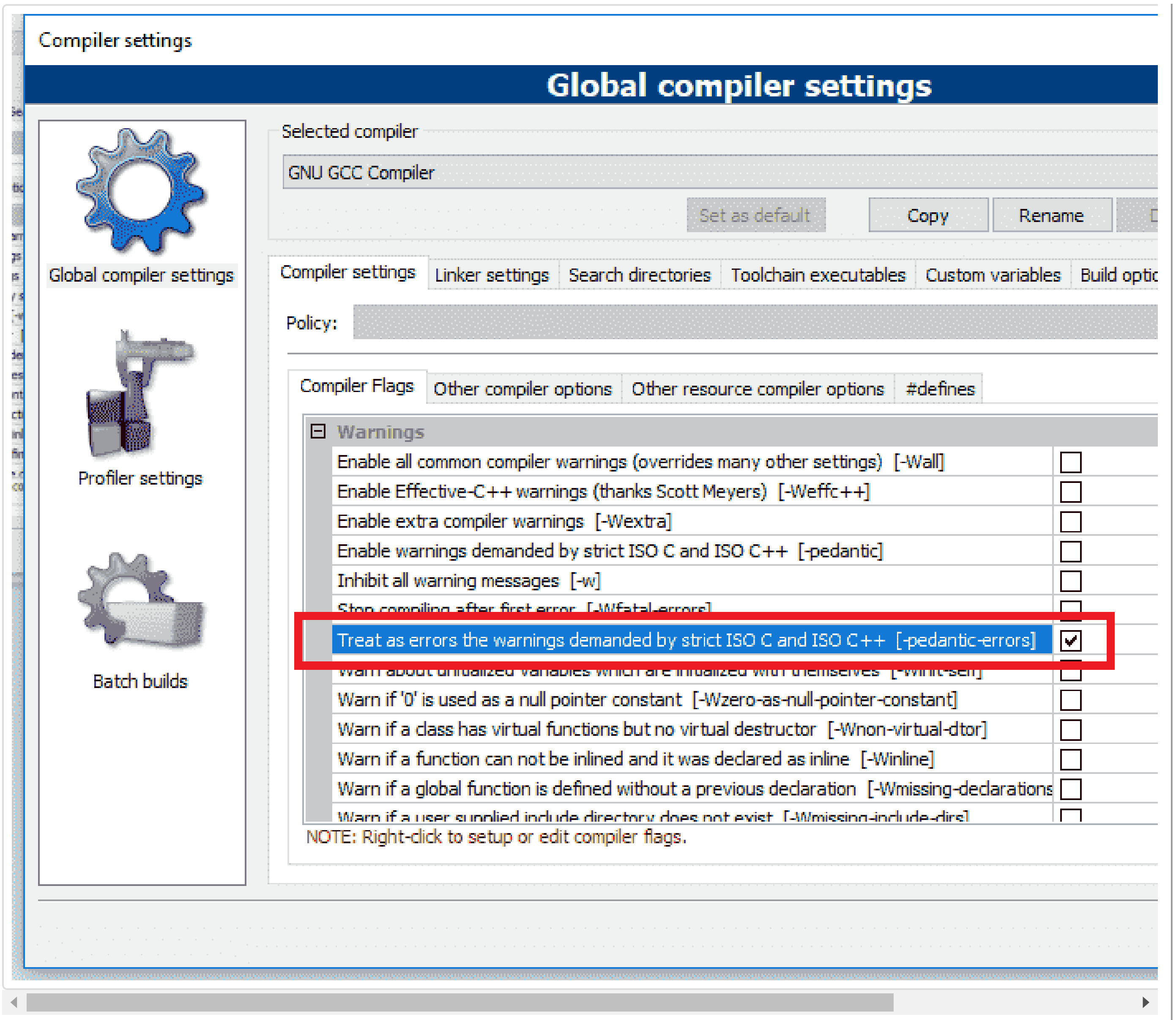
Platform: Active(Win32)

Disable Language Extensions	**Yes (/Za)**
Conformance mode	**Yes (/permissive-)**
Treat WChar_t As Built in Type	Yes (/Zc:wchar_t)
Force Conformance in For Loop Scope	Yes (/Zc:forScope)
Remove unreferenced code and data	Yes (/Zc:inline)
Enforce type conversion rules	
Enable Run-Time Type Information	
Open MP Support	
C++ Language Standard	
Enable C++ Modules (experimental)	

Disable Language Extensions
Suppresses or enables language extensions. (/Za)

For Code::Blocks users

Disable compiler extensions via *Settings menu* > *Compiler* > *Compiler flags tab*, then find and check the *-pedantic-errors* option.



- [ ] Enable all common compiler warnings (overrides many other settings) [-Wall]
- [ ] Enable Effective-C++ warnings (thanks Scott Meyers) [-Weffc++]
- [ ] Enable extra compiler warnings [-Wextra]
- [ ] Enable warnings demanded by strict ISO C and ISO C++ [-pedantic]
- [ ] Inhibit all warning messages [-w]
- [ ] Stop compiling after first error [-Wfatal-errors]
- [✓] Treat as errors the warnings demanded by strict ISO C and ISO C++ [-pedantic-errors]
- [ ] Warn about uninitialized variables which are initialized with themselves [-Winit-self]
- [ ] Warn if '0' is used as a null pointer constant [-Wzero-as-null-pointer-constant]
- [ ] Warn if a class has virtual functions but no virtual destructor [-Wnon-virtual-dtor]
- [ ] Warn if a function can not be inlined and it was declared as inline [-Winline]
- [ ] Warn if a global function is defined without a previous declaration [-Wmissing-declarations
- [ ] Warn if a user supplied include directory does not exist [-Wmissing-include-dirs]

NOTE: Right-click to setup or edit compiler flags.

For GCC/G++ users

You can disable compiler extensions by adding the *-pedantic-errors* flag to the compile command line.

Related content

Xcode users can refer to **Rory's comment**, who kindly provided instructions.

A reminder

These settings are applied on a per-project basis. You need to set them every time you create a new project, or create a template project with those settings once and use that to create new projects.

0.11 -- Configuring your compiler: Warning and error levels

Index

0.9 -- Configuring your compiler: Build configurations

C++ TUTORIAL | PRINT THIS POST

60 comments to 0.10 — Configuring your compiler: Compiler extensions

Warteks
January 31, 2020 at 8:37 pm · Reply

I'm posting this here since its the first part in the tutorial that suggests opening project properties in Visual Studio. I had a strange situation, and googled and googled with no answers so I tried my own solution and it worked.

I simply could NOT open the properties window for the project. I just upgraded my Visual Studio to

Microsoft Visual Studio Community 2017 Version 15.9.19
Microsoft .NET Framework Version 4.7.02558
Visual C++ 2017   00369-60000-00001-AA098

on Windows 7 Ultimate SP1 6.1.7601

Anyways, properties simply wouldn't do anything, no error, no window, no nothing. But in the corner forced properties box I would try to click on stuff and get a error "Font 'Calibri Bold Caps' can not display regular" or something close to that. I got old google results for similar message but other windows software and different font types.

The solution: deleted the font after closing all associated programs. Now everything works fine. I'm sure this is just some weird glitch from another fantastic Microsoft product update. The file was "calibri bold caps_0.ttf" and I'm certain I wont miss it, the rest of the calibri family are still there.

ssn26989
January 27, 2020 at 11:02 pm · Reply

Hi dear nascardriver
Finally I upgraded my compiler to c++17.
Problems were solved by installing the latest version of MinGW and making the necessary changes to the compiler, but in Visual Studio 2019 the standard language definition for each project have to set separately.
Is there any way to do this at once for all new projects?
Thanks for your help.

nascardriver
January 28, 2020 at 2:27 am · Reply

Hi!

No, not directly. You can create a project with all the recommended settings (Standard, Warnings, Extensions) and save that project as a template. Then, when you create a new project, create the project from that template. Google should find you instructions on how to do it.

ssn26989
January 25, 2020 at 1:11 am · Reply

Hi dears
Why my code has error only for line number seven.
My compiler is tdm64-gcc-5.1.0-2.c++17.
This compiler appears to support c++17.
Where is my mistake. Please help me.
Thank you for your help.

```
1   #include <array>
2   #include <iostream>
3
4   int main()
5   {
6       int ws{12};
7       std::array myArray { 9.0, 7.2, 5.4, 3.6, 1.8 };
8       std::cout << "ws is: " << ws << "\n";
9       return 0;
10  }
```

Compiler error:

```
x86_64-w64-mingw32-g++.exe -pedantic-errors -Wall -std=c++17 -fexceptions -std=c++17 -g -pedantic-errors -std=c++17  -c "D:\My Documents\C++ Source\Code Blocks\Project 098\main.cpp" -o obj\Debug\main.o
x86_64-w64-mingw32-g++.exe  -o "bin\Debug\Project 098.exe" obj\Debug\main.o
D:\My Documents\C++ Source\Code Blocks\Project 098\main.cpp: In function 'int main()':
D:\My Documents\C++ Source\Code Blocks\Project 098\main.cpp:7:16: error: missing template arguments before 'myArray'
    std::array myArray { 9.0, 7.2, 5.4, 3.6, 1.8 };
               ^
```

nascardriver
January 25, 2020 at 3:29 am · Reply

gcc 5 has virtually no support for C++17, see https://gcc.gnu.org/projects/cxx-status.html#cxx17 . You need at least gcc 7.

Rory
January 2, 2020 at 12:01 pm · Reply

For those asking about Xcode, here is how you add compiler flags:

1. In Xcode, press CMD+1 to show the Project Navigator.
2. In the Project Navigator click on your project, the one with the blue file icon.
3. In the main editor window, select the Target.
4. Click on Build Settings.
5. Make sure "All" and "Combined" are selected.
6. Type "c++ flag" in the search filter.
7. You should now see the setting "Other C++ Flags" under the header "Custom Compiler Flags".
8. Double-click to the right of that line. A small box pops up.

9. Click the "+" button, type in "-std=c++17" and press Enter to save.
10. Click "+" again, type in "-pedantic-errors" and hit Enter.

You're done! Click out to dismiss the popup and you'll see both flags have been added to the "Other C++ Flags" setting. You could also have single-clicked on the right and instead of the popup it turns into an editable text field. You can add flags this way too just by typing them all in one space-separated line. Just leave a space after $(OTHER_CFLAGS) and add your flags at the end. Don't remove $(OTHER_CFLAGS)!

Note 1:
In Xcode 11.3 Build Settings, under the Warning Policies header you'll see settings for "Pedantic Warnings" and "Treat Warnings as Errors". Setting these both to "Yes" however, only seems to add the "-pedantic" flag. Not "-pedantic-errors" as we want. Not sure if bug or intended so for now I guess leave these both at "No" and use the Custom Compiler Flags setting as shown above.

Note 2:
There is a Build Setting called "C++ Language Dialect". Just type in "dialect" in the search filter and you'll find it. You can click the drop-down and select C++17 here. This will send the flag "-std=c++1z" to the compiler so you don't have to manually add it to the custom flags setting.

Edit: Found out C++1z was the name of the draft version of C++17 before it was approved.

nascardriver
January 3, 2020 at 5:54 am · Reply

Thank you for the instruction! Many readers asked about Xcode, I'm sure they'll appreciate your efforts. I linked your comment in the lesson to make it easier to find.

Rory
January 3, 2020 at 2:24 pm · Reply

No problem! Hope it helps.

Vitaliy Sh.
December 25, 2019 at 3:09 am · Reply

man g++:
"Most of the command-line options that you can use with GCC ..."

here:
"You can disable compiler extensions by adding the -pedantic-errors flag to the compile command line." -- "compileR" ?

Maybe your change like:

```
<p>You can disable compiler extensions by using the <em>-pedantic-errors</em> flag on the c
ommand line: <code>g++ -pedantic-errors</code><br>
</p>
```

?

Vitaliy Sh.
December 25, 2019 at 2:36 am · Reply

web.printRand("HeLooooovesN*****Cards");

"There are competent people on standards bodies. But they aren't
always competent, and the translation of intent to English isn't

always perfect either. So standards are not some kind of holy book
that has to be revered. Standards too need to be questioned." (the C language)

https://lkml.org/lkml/2018/6/5/769

There were other citations as well :)

Shinawil
October 5, 2019 at 3:56 pm · Reply

Xcode question/solution to turning off C++ extensions:
Will this cover me in build settings key value pairs? 'Using C++ extensions in earlier versions of C++'? 'No'.

James O Real
September 21, 2019 at 12:21 am · Reply

Honestly, this does not answer the question for those using xcode. An explicit example should be provided to address the lesson. Simply telling people to continue to the next lesson is bypassing the entire reason they are here.

SRM
September 10, 2019 at 7:31 am · Reply

For anyone who is wondering how to use XCode this answer on StackOverflow seems promising
https://stackoverflow.com/questions/37668204/disable-c-optimisation (2nd answer)
It doesn't say whether it removes all compiler extensions, but this is apparently how you do it.
Also this is on a file-by-file basis, but I guess at least for a couple of chapters this is likely only main.cpp anyway, so one file to add flags for

nascardriver
September 10, 2019 at 7:34 am · Reply

Nope, that's disabling optimization. Optimization is good.

James O Real
September 21, 2019 at 12:27 am · Reply

@nascardriver, if you know the solution for those using xcode, it would be a big help if you just provided it.

nascardriver
September 21, 2019 at 1:59 am · Reply

I never used XCode, I don't know. Afaik XCode is a text editor, not an IDE. If that's the case, check which compiler you're using and look up the options of that compiler.

Wallace
October 3, 2019 at 3:13 pm · Reply

I'm only learning Xcode, using v. 10.2, but by searching for "pedantic," as shown for Code::Blocks, in Xcode's help, I find the following in the "Build settings reference" (https://help.apple.com/xcode/mac/10.2/index.html?localePath=en.lproj#/itcaec37c2a6):

"Pedantic Warnings (GCC_WARN_PEDANTIC)
"Issue all the warnings demanded by strict ISO C and ISO C++; reject all programs that use forbidden extensions, and some other programs that do not follow ISO C and ISO C++. For ISO C, follows the version of the ISO C standard specified by any -std option used."

That's not as strong as elevating all warnings to errors, but it appears to be the closest option to this tutorial's intent.

Another relevant option: "C Language Dialect (GCC_C_LANGUAGE_STANDARD)" and the similar but undocumented C++ language dialect.

To set these, see Xcode help titled "Configure build settings" (https://help.apple.com/xcode/mac/10.2/index.html?localePath=en.lproj#/dev04b3a04ba). In the GUI, you can search for "pedantic" and "dialect" to go straight to these settings.

nomnom
November 6, 2019 at 6:50 am · Reply

Perhaps you could try using Clang (another compiler)? It supports all 3 major platforms (Mac, Windows and Linux). It's support is fairly extensive and is more or less on par with others in terms of optimized code. Speaking of IDEs, 'Visual Studio for Mac' is out, if you're interested

Rory
January 2, 2020 at 12:53 pm · Reply

Visual Studio for Mac only supports C# at this time. It does not yet support C++.

Kasra
August 23, 2019 at 8:10 am · Reply

Hello!
I read this topic but i'm using Dev-C++ and you didn't mention this compiler in your post.Should I change anything in this compiler for better configuration?
Thanks for your answer... :)

nascardriver
August 23, 2019 at 8:19 am · Reply

Dev-C++ isn't a compiler, it's an IDE.
Based on my google search, dev-c++ is still compiling with gcc-4.9. It's long outdated. See if there's an up-to-date version of dev-c++ or search for a new IDE.
You can follow the GCC/G++ guide, you just have to find the options in your IDE.

Dqueezy
August 22, 2019 at 3:02 pm · Reply

Do we have to disable those extensions every time we open Visual Studio or can it be done permanently?

nascardriver
August 23, 2019 at 2:04 am · Reply

They're applied on a per-project basis. You can create a project, apply the settings, and save the project as a template.

Beckett
July 1, 2019 at 10:03 am · Reply

Hey Alex!
Great tutorial so far, something to add possibly
Later on when we're talking about explicit vs implicit initialization, I ran into a problem where
> int x {5};
popped up an error because i had turned the sensitivity of the errors up but hadn't allowed the compiler to utilize C++11.

I did find a fix for it though

1 Go to Toolbar -> Settings -> Compiler
2 In the Selected compiler drop-down menu, make sure GNU GCC Compiler is selected
3 Below that, select the compiler settings tab and then the compiler flags tab underneath
4 In the list below, make sure the box for "Have g++ follow the C++11 ISO C++ language standard [-std=c++11]" is checked
5 Click OK to save

My question is, for this tutorial should we be using C++, C++11, or C++14?

-me

Alex
July 1, 2019 at 1:05 pm · Reply

IMO, you should use the highest non-experimental version that your compiler supports (C++17 as of right now). If your compiler doesn't support C++17, then C++14 as a fallback.

oxygène
April 8, 2019 at 11:57 pm · Reply

any specific configuration required for Kdevelop c++ cmake terminal project?

nascardriver
April 9, 2019 at 12:13 am · Reply

There should be a CMakeLists.txt file in your project.

```
// This should already be in CMakeLists.txt
project(your_project_name VERSION 1.0.0)

// ...

// Add this
```

```
set(CMAKE_CXX_STANDARD "20")

// ...

// This should already be there. Variable names might differ
add_executable(${PROJECT_NAME} ${SOURCE})

// ...

// And add these
target_compile_options(${PROJECT_NAME} PUBLIC "-Wall")
target_compile_options(${PROJECT_NAME} PUBLIC "-Wextra")
target_compile_options(${PROJECT_NAME} PUBLIC "-Weffc++")
target_compile_options(${PROJECT_NAME} PUBLIC "-Wconversion")
target_compile_options(${PROJECT_NAME} PUBLIC "-pedantic-errors")
```

oxygène
April 27, 2019 at 3:44 am · Reply

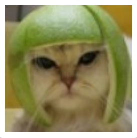
Thanks. Does it also work for qt creator?

nascardriver
April 27, 2019 at 3:51 am · Reply

I think qt uses cmake or a variant of it. The configuration process should be the same or similar, you'll have to look it up on your own.

Matt
March 11, 2019 at 6:08 pm · Reply

Does Visual Studio 2017 have a default setting to disable these extensions permanently? I haven't been able to find it myself, but that's not saying much.



Alex
March 11, 2019 at 10:29 pm · Reply

Not that I've found, which is pretty unexpected.

nascardriver
March 13, 2019 at 4:51 am · Reply

Hi Matt!

You can create a project template and use that whenever you create a new project.
https://docs.microsoft.com/en-us/visualstudio/ide/how-to-create-project-templates?view=vs-2017

Ferd Burfel
March 6, 2019 at 1:23 pm · Reply

Alex, I would recommend for VS2017 users you mention the conformance mode (/permissive-) just below where language extensions can be disabled. It is on by default, so VS can check if the C++ conforms to the selected C++ standard (C++14/C++17/C++latest).

Alex
March 8, 2019 at 4:05 pm · Reply

I don't mention it because it's on by default. Is there some specific reason you think it's worth calling out explicitly?

Bebeji
January 20, 2019 at 8:13 am · Reply

Using Xcode Version 10.1 (10B61). Not sure how to turn off compiler extensions here. Any idea?

Lachie
December 24, 2018 at 4:16 am · Reply

Thanks for the tutorials I am loving them so far but I am using CLion and MinGw to compile and can't find a similar setting can anyone assist me to find the setting or is it disabled by default in CLion?

Thanks, Lachie

sasaki
December 6, 2018 at 6:29 am · Reply

i got most of it. i mean i hope i did. but i still don't get how to write all those coding in each line.

Elijah
November 21, 2018 at 11:48 pm · Reply

almost none of the options in HelloWorld project properties shown in your screenshots appear in mine. am I dong something wrong? did I not install the right package?

all that pops up is
Common properties
    Startup Project
    Project Dependencies
    Code Analysis Settings
    Debug Source Files
Configuration Properties
    Configuration

also, I can't seem to set the platform to active(win32)

thank you if anyone gets around to helping me out

best regards

Alex
November 27, 2018 at 12:44 pm · Reply

Were you able to create your project as a win32/windows console project in the first place?

Also make sure you're right clicking on the project line in the solution explorer (in the screenshot, the one that says "ConsoleApplication9"), not the solution line (in the screenshot, the one that says "Solution ConsoleApplication9 (1 project)")

Cas
November 15, 2018 at 6:15 pm · Reply

For mac users we were told to get xcode as our compiler but now there is no instruction for it. Please help. I came to learn C++ but now can't get through Chapter 0.

nascardriver
November 16, 2018 at 12:14 am · Reply

The steps in this lesson are highly recommend, though not mandatory. You can continue to the next lesson.

Quackky
November 11, 2018 at 11:14 pm · Reply

I just finished disabling language extensions for the HelloWorld project. Do I have to do it again every time for future projects?

nascardriver
November 12, 2018 at 9:46 am · Reply

Unless your IDE has a way of changing the default settings, yes, you have to do it for every new project.

Dave
November 9, 2018 at 8:20 pm · Reply

Got the extensions disabled (Codeblocks) but there was another option checked already which was:
Have g++ follow the C++ I4ISOC++ language standard[std=C++I4]
Should that be left checked?
Sorry if is a dumb question but I know nothing about working with a compiler and only have used perl and php and want to have as much right as possible.

nascardriver
November 10, 2018 at 8:47 am · Reply

If that's the newest supported version, yes. If there's a newer version of C++ available in your compiler, use that.

Dave
November 10, 2018 at 10:33 am · Reply

Thanks, I just downloaded the compiler codeblocks-17.12mingw-setup.exe from Sourceforge. When disabling the extensions was when I noticed.
Have a great day :)

Leo

October 24, 2018 at 3:08 pm · Reply

How can i disable the complier extensions on Xcode? I have searched sound a bit and can't find anything on it. Is there something better I could be using on a mac? I could even install windows on another portion or something it it would be easier to use windows. What do you think?

Qluefqen
October 20, 2018 at 4:21 am · Reply

Hi, Alex,

In keeping with your advice to experiment, I've tried using this command-line compiler line:

```
g++ -Wall -pedantic-errors -std=c++17 -o <filename>.app <filename>.cpp
```

That works fine with g++, but when I try any variant of it with gcc, I get:

```
/usr/bin/ld: /tmp/ccWeSps4.o: in function `main':
hello.cpp:(.text+0xe): undefined reference to `std::cout'
/usr/bin/ld: hello.cpp:(.text+0x13): undefined reference to `std::basic_ostream<char, std::char_traits<char> >& std::operator<< <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char> >&, char const*)'
/usr/bin/ld: hello.cpp:(.text+0x1d): undefined reference to `std::basic_ostream<char, std::char_traits<char> >& std::endl<char, std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char> >&)'
/usr/bin/ld: hello.cpp:(.text+0x28): undefined reference to `std::ostream::operator<<(std::ostream& (*)(std::ostream&))'
/usr/bin/ld: /tmp/ccWeSps4.o: in function `__static_initialization_and_destruction_0(int, int)':
hello.cpp:(.text+0x58): undefined reference to `std::ios_base::Init::Init()'
/usr/bin/ld: hello.cpp:(.text+0x6d): undefined reference to `std::ios_base::Init::~Init()'
```

Also, I found that this works best for me and would ask for your comments:

```
#include <iostream>

int main()
{
    std::cout << "The sum of 5 and 7 is "
              << 5+7
              << "."
              << std::endl;

    return 0;
}
```

I just find it easier to read and my compiler seems to have no problem with me not putting the semicolon at the end of every line. Am I screwing up my learning in any way?

nascardriver
October 20, 2018 at 4:26 am · Reply

Hi!

gcc compiles C, g++ compiles C++.

> Am I screwing up my learning in any way?
Nope, that's fine.

Hiruna

September 28, 2018 at 8:59 pm · Reply

Alex when I build solution after I give Yes(/Za) to disable language extentions in visual studio 2017 , It show an error, in it they say that (/Za) is an invalid value to disable language extentions

Alex
October 1, 2018 at 8:16 am · Reply

Can you share the exact error message you're getting?

bibek
September 22, 2018 at 10:49 pm · Reply

i really could not understand how to disable compiler extensions in G++
CAN U PLEASE CLARIFY IT

nascardriver
September 23, 2018 at 1:09 am · Reply

```
g++ -pedantic-errors ./main.cpp
```

Khang
September 20, 2018 at 2:44 am · Reply

Hi Alex,

If I disabled the compiler extension in Visual Studio 2015, when I use a header guard and include it inside my main file:
Header file (header.h):

```
#ifndef SOMEHEADER_H
#define SOMEHEADER_H

// some code here

#endif  // will get error C1004 : unexpected end-of-file found
```

Main file (main.cpp):

```
#include <iostream>
#include "header.h"

int main()
{
    return 0;
}
```

The compiler will throw an error telling me that #endif is illegal, but if I delete it, it will tell me that #endif is missing. Can you please tell me what is the problem here? Thank you.

Alex
September 20, 2018 at 8:47 am · Reply

I've having no issues compiling your code with compiler extensions disabled (substituting in a template class for // some code here)

If you remove all of the "// some code here", does it work?
Does it work if you disable compiler extensions again?

Khang
September 20, 2018 at 10:29 pm · Reply

I think I have found the problem, it seems like when I disable the compiler extension, I must press enter after the "#endif" preprocessor to make a newline.

```
#ifndef HAHA_H
#define HAHA_H

#endif   //press enter here
//must have an empty line here, not even a comment can be put here
```

If I enter a newline, the program will compile, and vice versa.

On the other hand, my "main.cpp" file doesn't need any empty line on Visual Studio 2015 with the compiler extension off, but when I tried it on C-free 5.0, it will cause an error and tell me it need a newline at the end. (The raw options for C-free compiler is "-g -DDEBUG -pedantic-errors")

```
#include <iostream>
#include "header.h"

int main()
{
   return 0;
}
// no need for newline here in VS2015, but needed in C-free
```

This have confused me a lot, if the compiler extension is unnecessary and potentially dangerous, why did they include it in the compiler in the first place?

nascardriver
September 21, 2018 at 1:15 am · Reply

Hi Khang!

TJ Seabrooks and Rakete1111 posted a nice explanation on stackoverflow (https://stackoverflow.com/a/72409/9364954).
Omitting the line feed at the end of a file could cause problems prior to C++11. Every C++11 and newer compiler should automatically add the line feed and not complain.

nascardriver
September 20, 2018 at 12:41 am · Reply

Hi Alex!

To my understanding -pedantic only disables extensions that prevent standard-conform programs from compiling and issues warnings when other extensions are used, but allows compilation. -pedantic-errors disables all extensions.

Alex
September 20, 2018 at 8:37 am · Reply

Thanks, I missed that nuance. Lesson updated!

Ahmed
September 19, 2018 at 2:45 pm · Reply

Thank you :)