0.11 — Configuring your compiler: Warning and error levels

BY ALEX ON SEPTEMBER 19TH, 2018 | LAST MODIFIED BY NASCARDRIVER ON DECEMBER 27TH, 2019

When you write your programs, the compiler will check to ensure you've followed the rules of the C++ language (assuming you've turned off compiler extensions, as per lesson **0.10 -- Configuring your compiler: Compiler extensions**).

If you have done something that definitively violates the rules of the language, during compilation the compiler will emit an **error**, providing both line number containing the error, and some text about what was expected vs what was found. The actual error may be on that line, or on a preceding line. Once you've identified and fixed the erroneous line(s) of code, you can try compiling again.

In other cases, the compiler may find code that seems like it might be in error, but the compiler can't be sure (remember the motto: "trust the programmer"). In such cases, the compiler may opt to issue a **warning**. Warnings do not halt compilation, but are notices to the programmer that something seems amiss.

Best practice

Don't let warnings pile up. Resolve them as you encounter them (as if they were errors).

In most cases, warnings can be resolved either by fixing the error the warning is pointing out, or by rewriting the line of code generating the warning in such a way that the warning is no longer generated.

In rare cases, it may be necessary to explicitly tell the compiler to not generate a particular warning for the line of code in question. C++ does not support an official way to do this, but many individual compilers (including Visual Studio and GCC) offer solutions (via non-portable #pragma directives) to temporarily disable warnings.

By default, most compilers will only generate warnings about the most obvious issues. However, you can request your compiler be more assertive about providing warnings for things it finds strange.

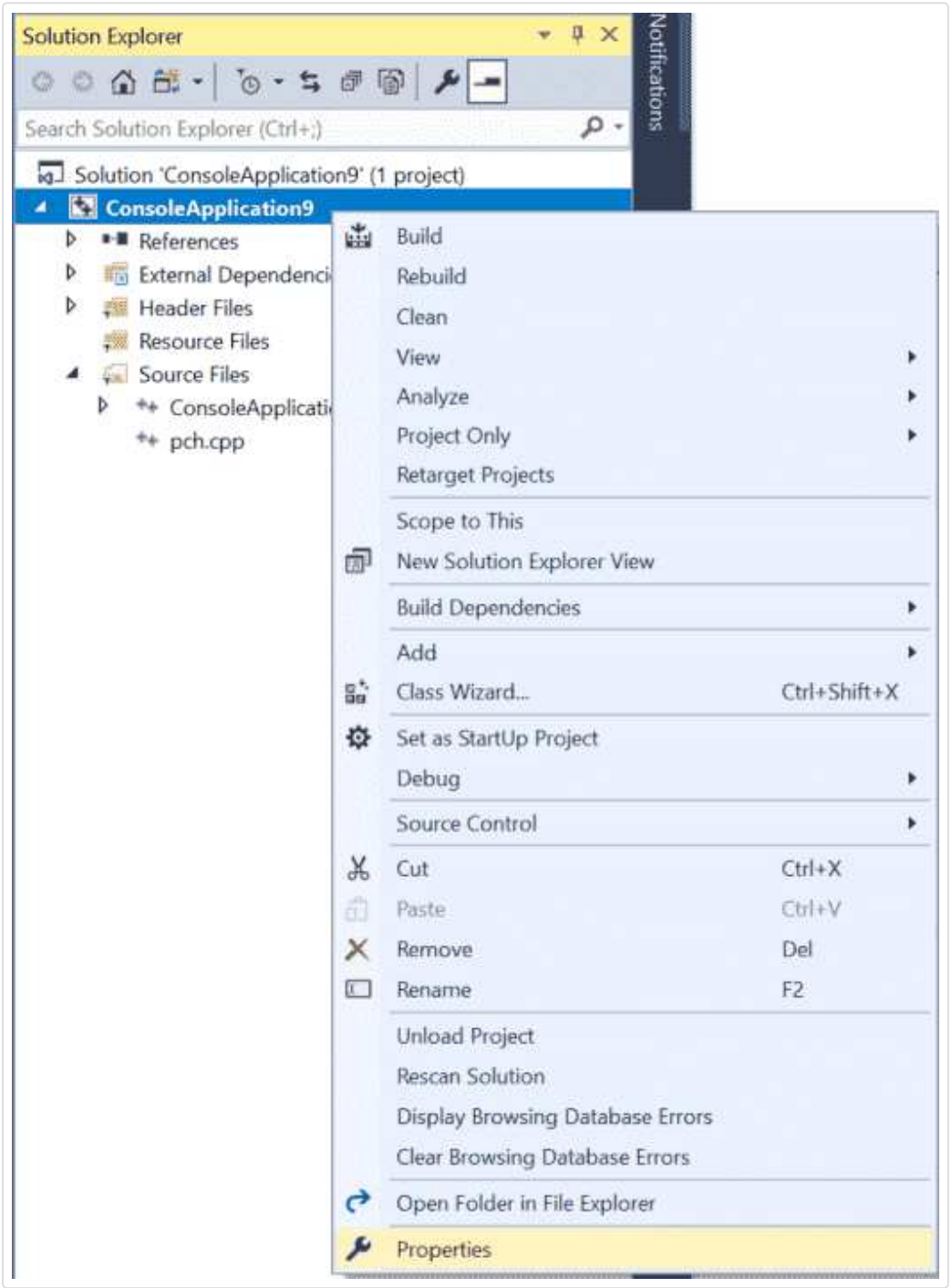
Best practice

Turn your warning levels up to the maximum, especially while you are learning. It will help you identify possible issues.

Increasing your warning levels

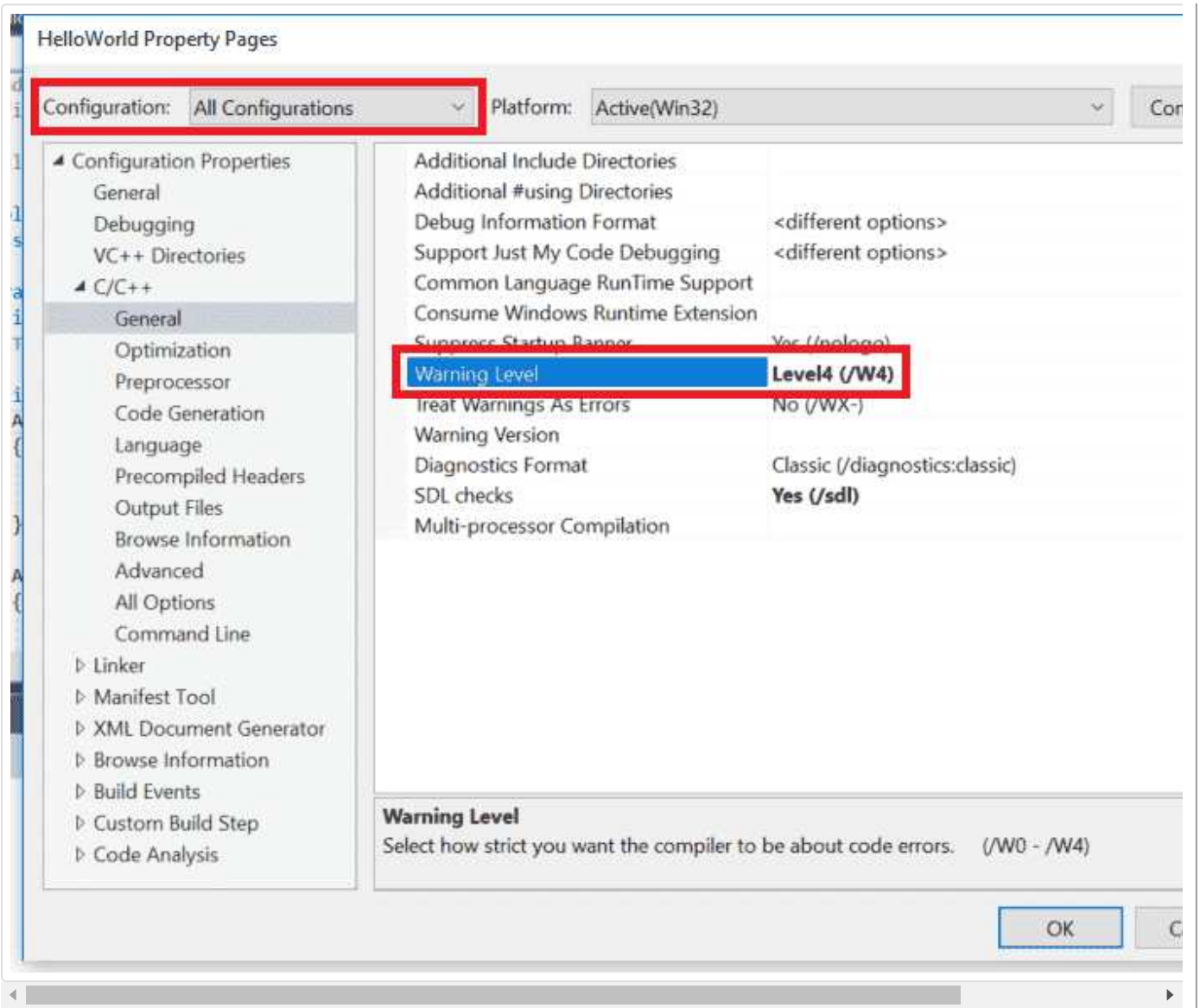
For Visual Studio users

To increase your warning levels, 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*.

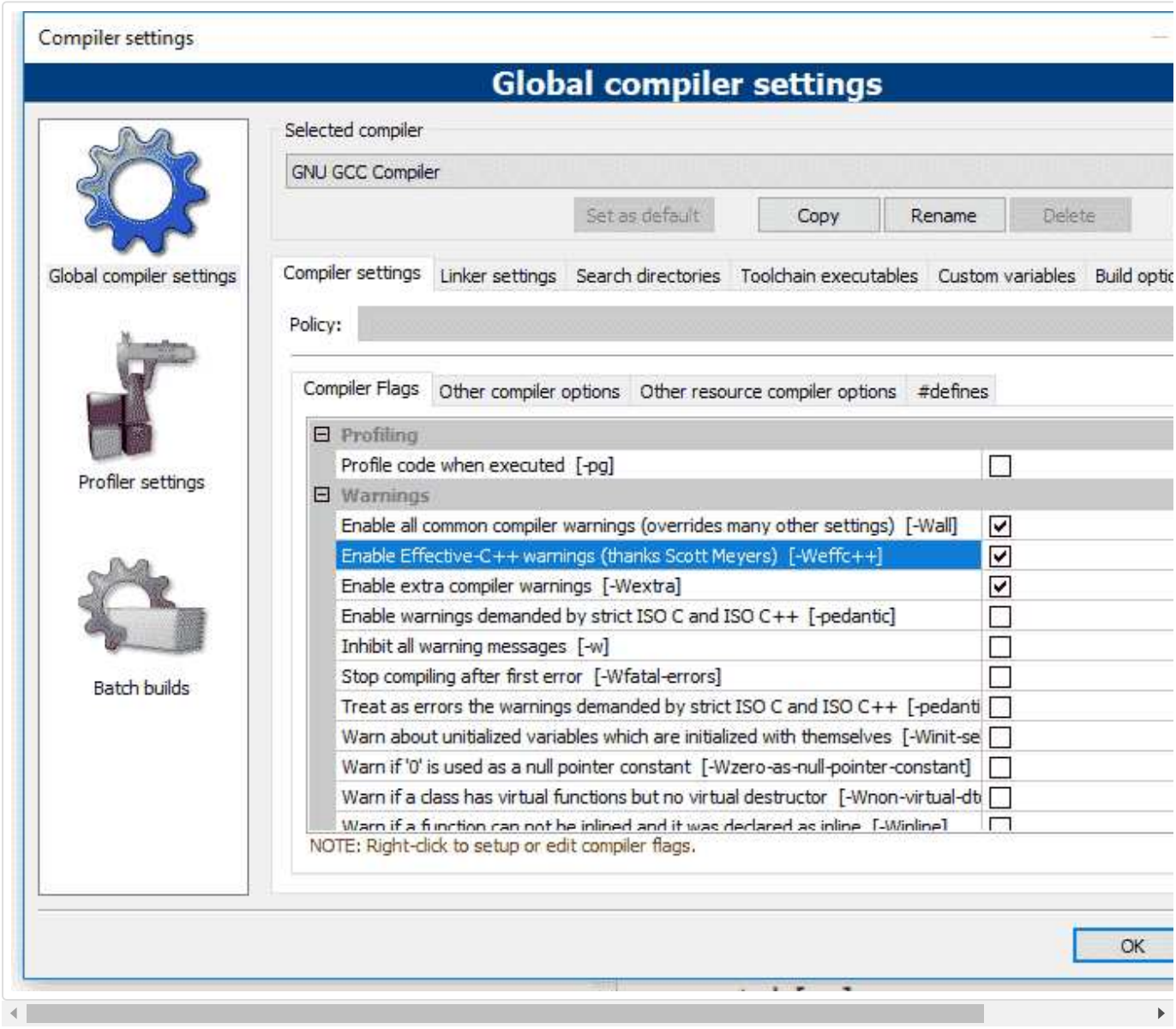
Then select *C/C++* > *General tab* and set *Warning level* to *Level4 (/W4)*:



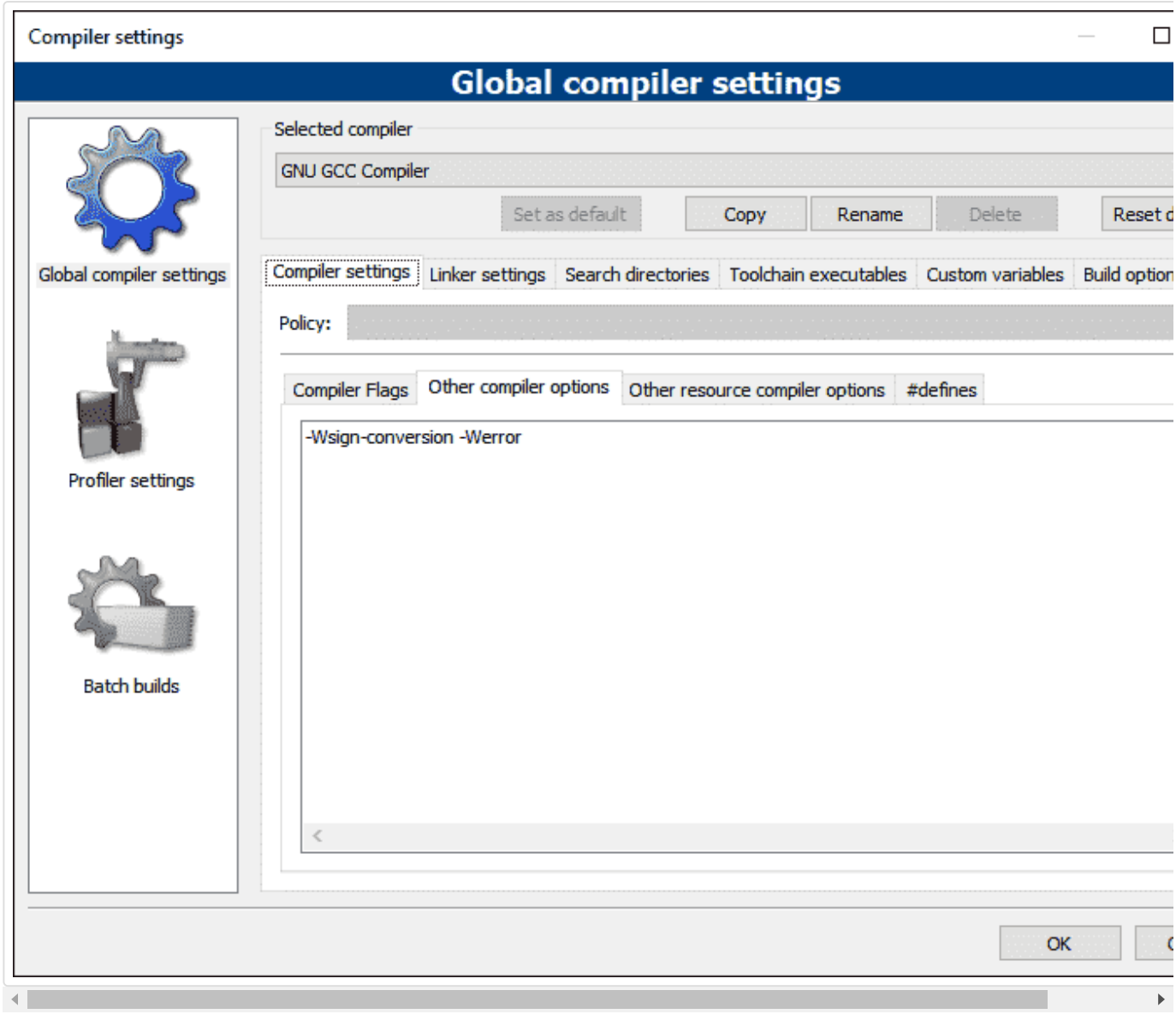
Note: Do not choose *EnableAllWarnings (/Wall)* or you will be buried in warnings generated by the C++ standard library.

For Code::Blocks users

From *Settings menu* > *Compiler* > *Compiler settings tab*, find and check the options that correlate with *-Wall*, *-Weffc++*, and *-Wextra*:



Then go to the *Other compiler options tab*, and add *-Wsign-conversion* to the text edit area:



Note: The *-Werror* parameter is explained below.

For GCC/G++ users

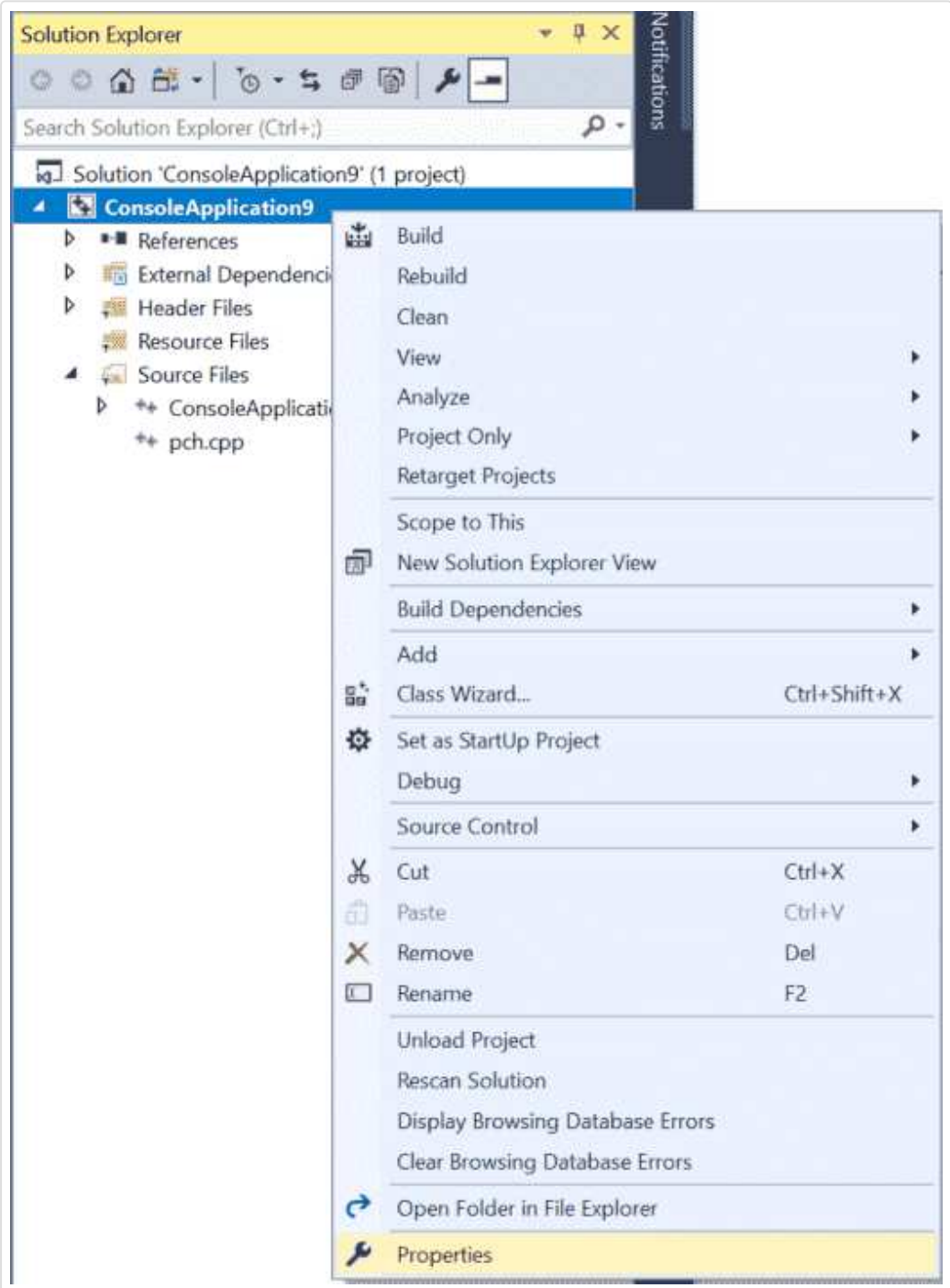
Add the following flags to your command line: *-Wall -Weffc++ -Wextra -Wsign-conversion*

Treat warnings as errors

It is also possible to tell your compiler to treat all warnings as if they were errors (in which case, the compiler will halt compilation if it finds any warnings). This is a good way to enforce the recommendation that you should fix all warnings (if you lack self-discipline, which most of us do).

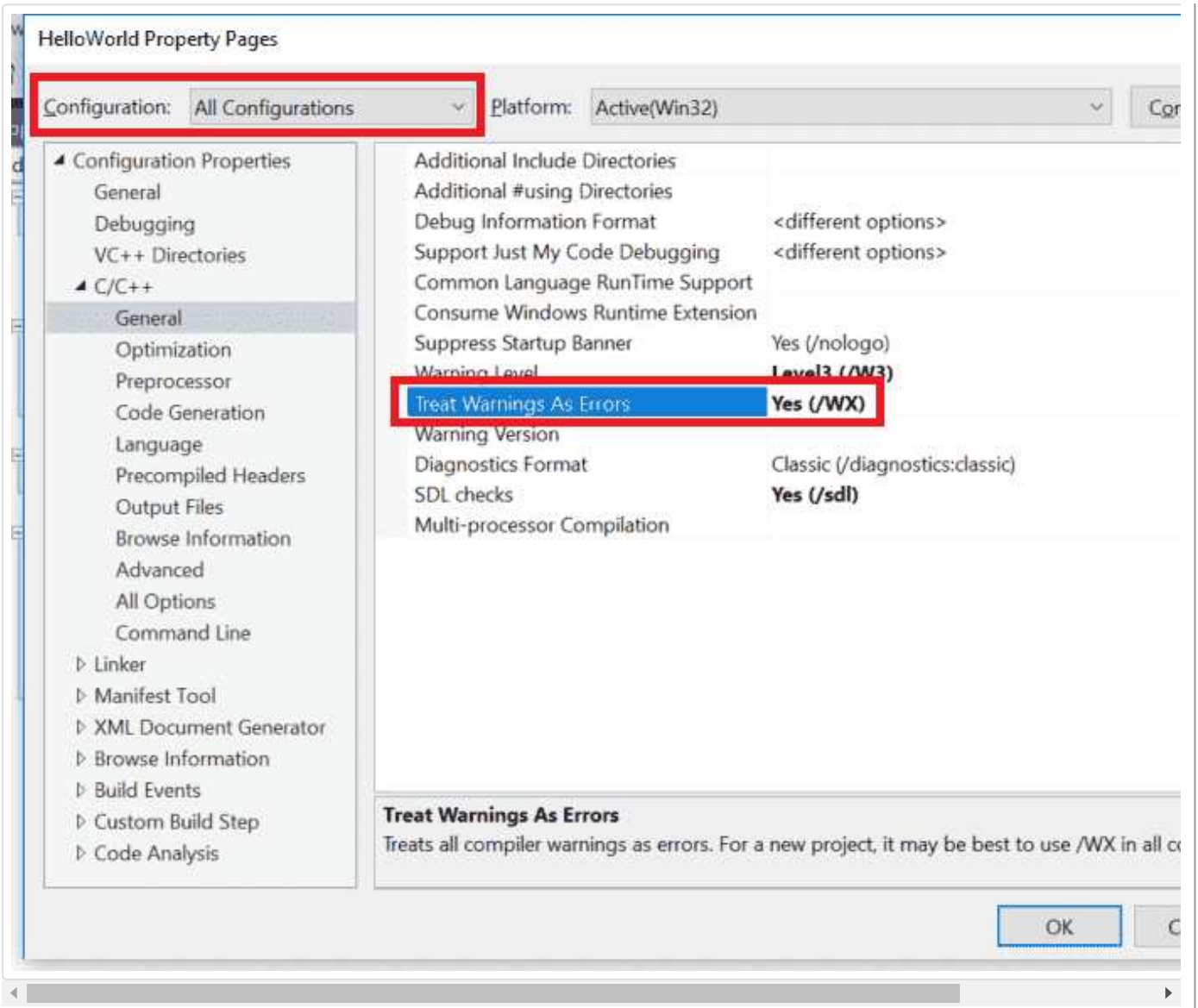
For Visual Studio users

To treat warnings as errors, right click on your project name in the *Solution Explorer* window, then choose *Properties*:



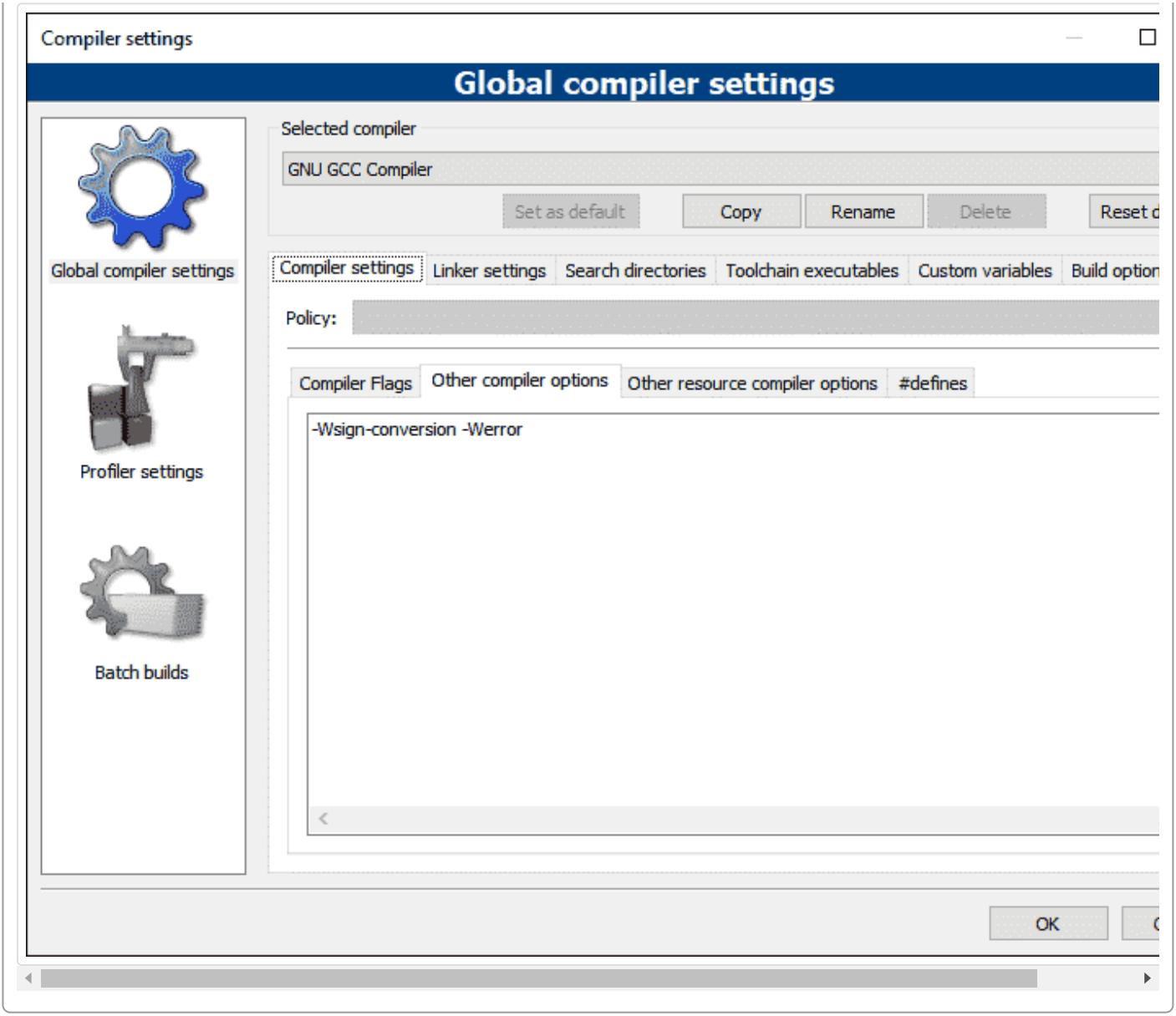
From the *Project* dialog, first make sure the *Configuration* field is set to *All Configurations*.

Then select *C/C++* > *General tab* and set *Treat Warnings As Errors* to *Yes (/WX)*.



For Code::Blocks users

From *Settings menu* > *Compiler* > *Other compiler options tab*, add *-Werror* to the text edit area:



For GCC/G++ users

Add the following flag to your command line: *-Werror*