0.11 — Configuring your compiler: Warning and error levels

learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/

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, the compiler is required to emit a **diagnostic message** (often called a **diagnostic** for short). The C++ standard does not define how diagnostic messages should be categorized or worded. However, there are some common conventions that compilers have adopted.

If compilation cannot continue due to the violation, then the compiler will emit an **error**, providing both line number containing the error, and some text about what was expected vs what was found. Errors stop the compilation from proceeding. 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.

For advanced readers

Syntax errors are always diagnosed as errors.

If compilation can continue despite the violation, the compiler may decide to emit either an error or a **warning**. Warnings are similar to errors, but they do not halt compilation.

In some cases, the compiler may identify code that does not violate the rules of the language, but that it believes could be incorrect. In such cases, the compiler may decide to emit a warning as a notice 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). Otherwise a warning about a serious issue may be lost amongst warnings about non-serious issues.

In most cases, warnings can be resolved either by fixing the issue 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.

Increasing your warning levels

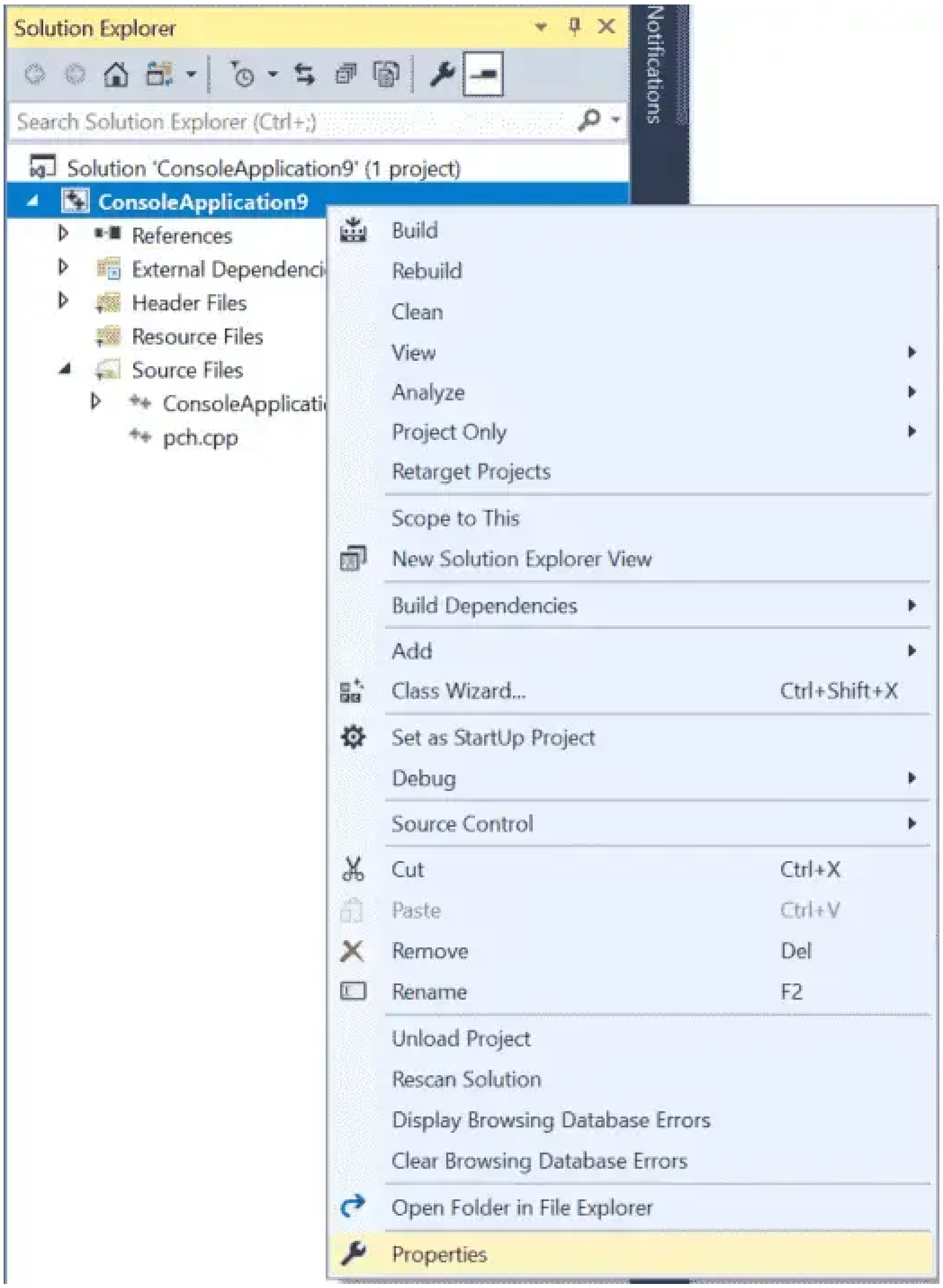
By default, most compilers will only generate warnings about the most obvious issues. However, you can request your compiler be more assertive about providing warnings for things it finds strange.

Best practice

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

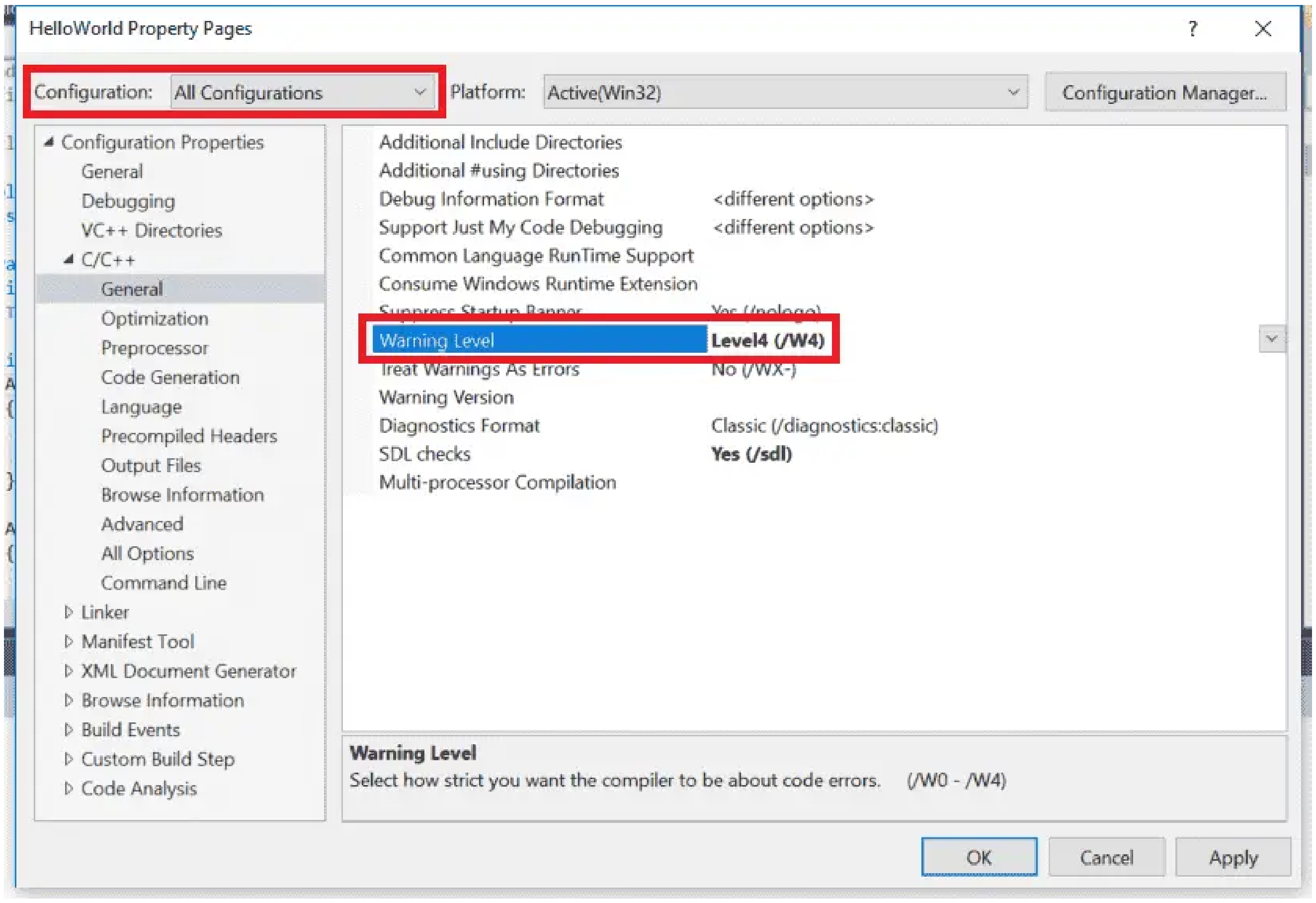
For Visual Studio users

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)*:



Additional Include Directories	
Additional #using Directories	
Debug Information Format	<different options>
Support Just My Code Debugging	<different options>
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	Level4 (/W4)
Treat Warnings As Errors	No (/WX-)
Warning Version	
Diagnostics Format	Classic (/diagnostics:classic)
SDL checks	Yes (/sdl)
Multi-processor Compilation	

Warning Level
Select how strict you want the compiler to be about code errors. (/W0 - /W4)

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

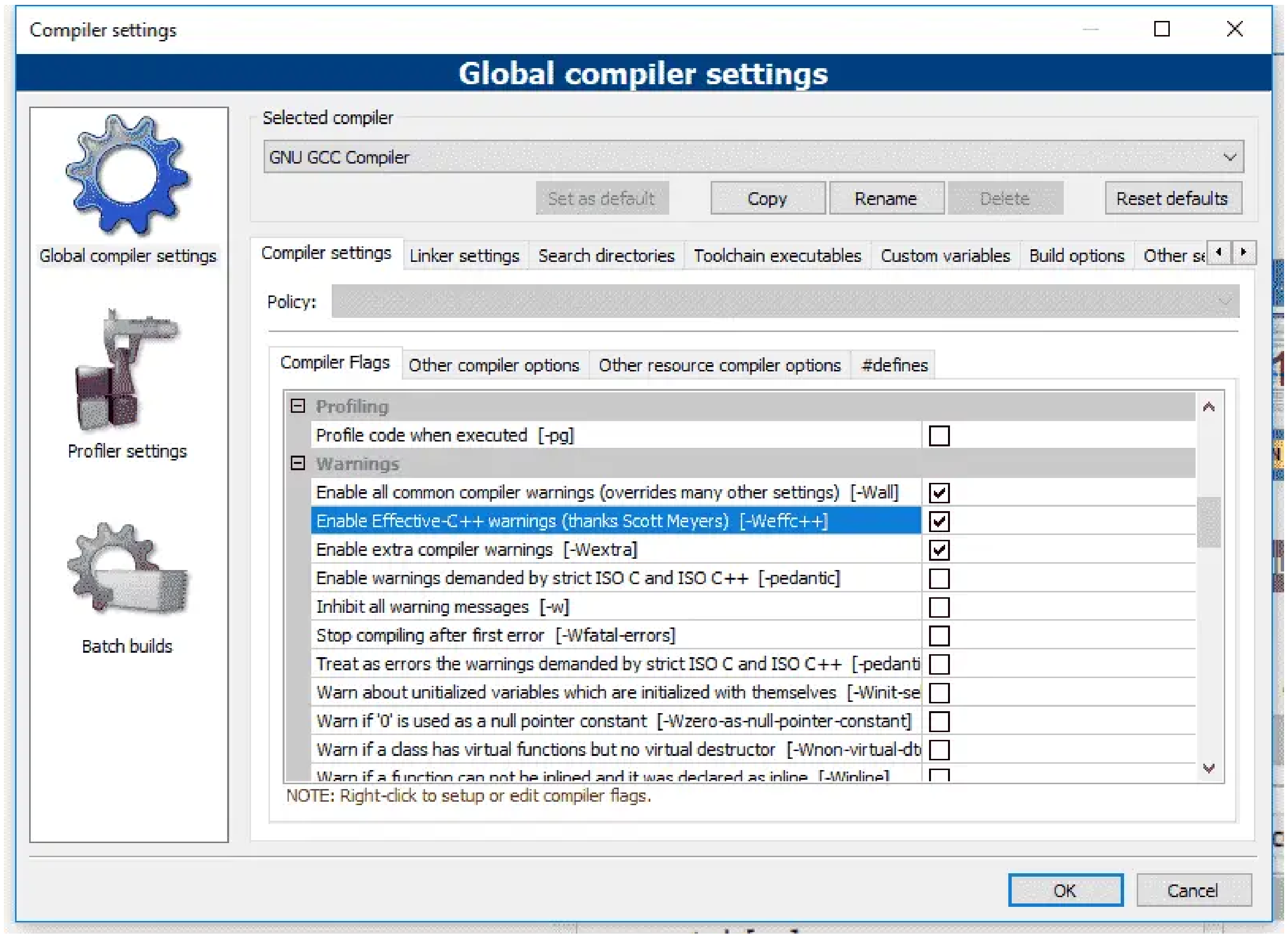
Visual Studio disables signed/unsigned conversion warnings by default, and those are useful, so if you are using Visual Studio 2019 or newer, let's enable those:

- From *C/C++ > Command Line tab*, under *Additional Options*, add `/w44365`. This tells the compiler to enable signed/unsigned conversion warnings at warning level 4 (which you enabled above).
- From *C/C++ > External Includes tab*, set *External Header Warning Level* to *Level3 (/external:W3)*. This tells the compiler to compile standard library headers at warning level 3 (instead of 4) so that compiling those headers doesn't trigger this warning.

The "External Includes" tab isn't shown in the graphic above, but appears in VS Community 2019 or newer between the "Browse Information" and "Advanced" tabs. See this link, which contains a recent photo of the dialog containing the "External Includes" tab.

For Code::Blocks users

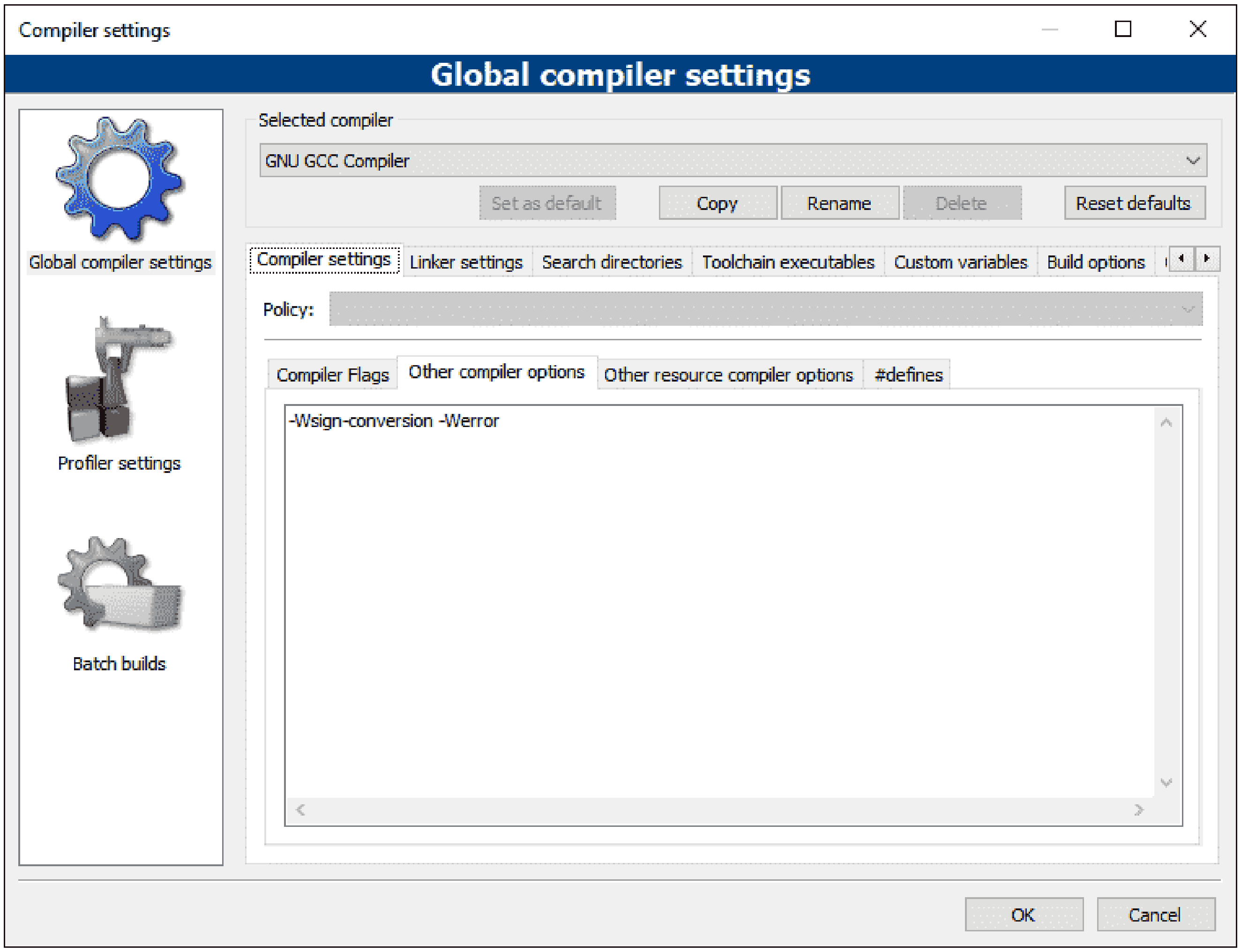
From *Settings menu > Compiler > Compiler settings tab*, find and check the options that correlate with *-Wall*, *-Weffc++*, and *-Wextra*:



⊟ Profiling	
Profile code when executed [-pg]	☐
⊟ Warnings	
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++ [-pedanti	☐
Warn about unitialized variables which are initialized with themselves [-Winit-se	☐
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-dt	☐
Warn if a function can not be inlined and it was declared as inline [-Winline]	☐

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

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



```
-Wsign-conversion -Werror
```

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

For GCC/G++ users

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

For VS Code users

Open the tasks.json file, find “args”, and then locate the line *“${file}”* within that section.

Above the *“${file}”* line, add new lines containing the following commands (one per line):

```
"-Wall",
"-Weffc++",
"-Wextra",
"-Wconversion",
"-Wsign-conversion",
```

Treat warnings as errors

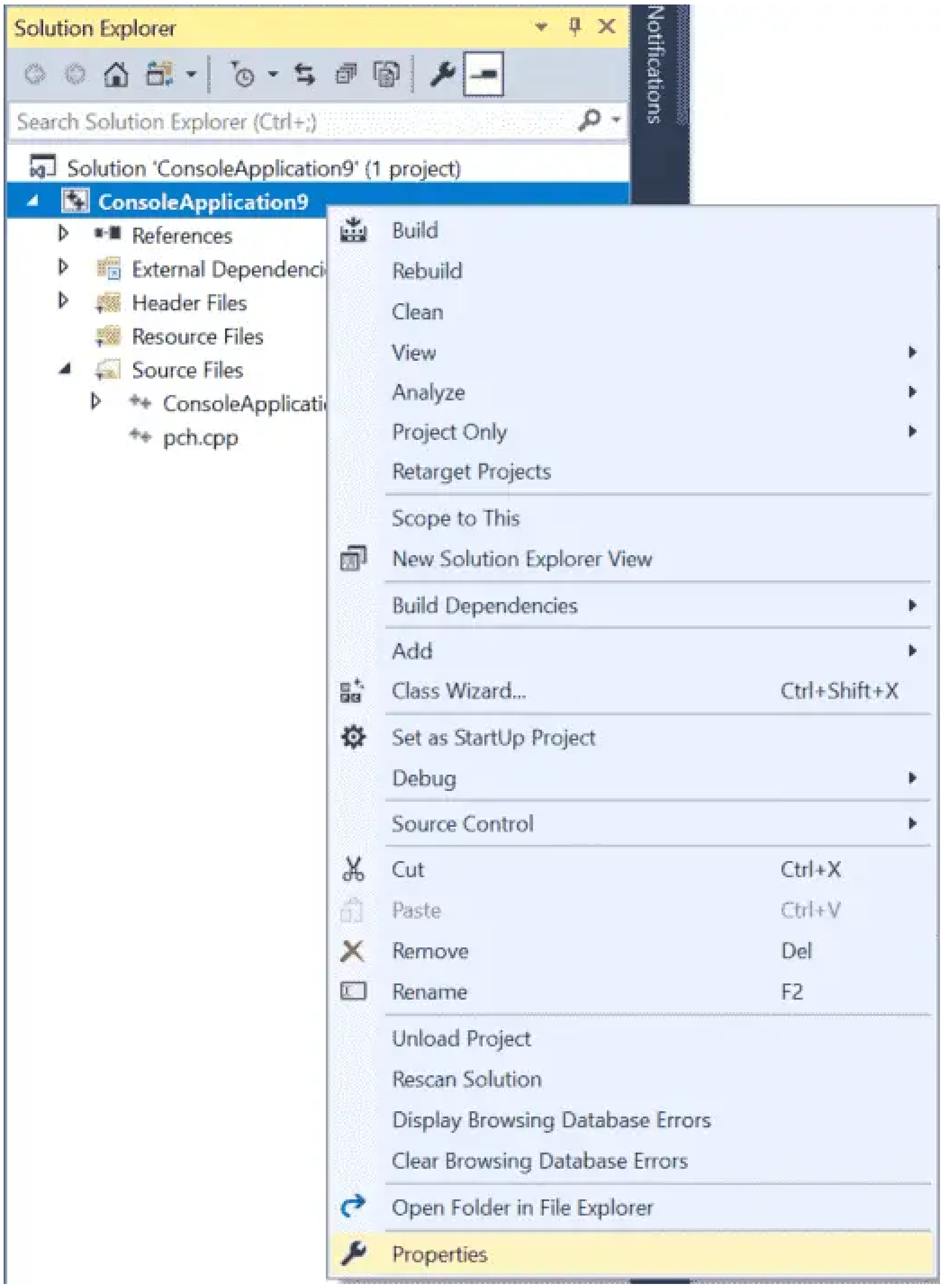
It is also possible to tell your compiler to treat all warnings as if they were errors (in which case, the compiler will halt compilation if it finds any warnings). This is a good way to enforce the recommendation that you should fix all warnings (if you lack self-discipline, which most of us do).

Best practice

Enable “Treat warnings as errors”. This will force you to resolve all issues causing warnings.

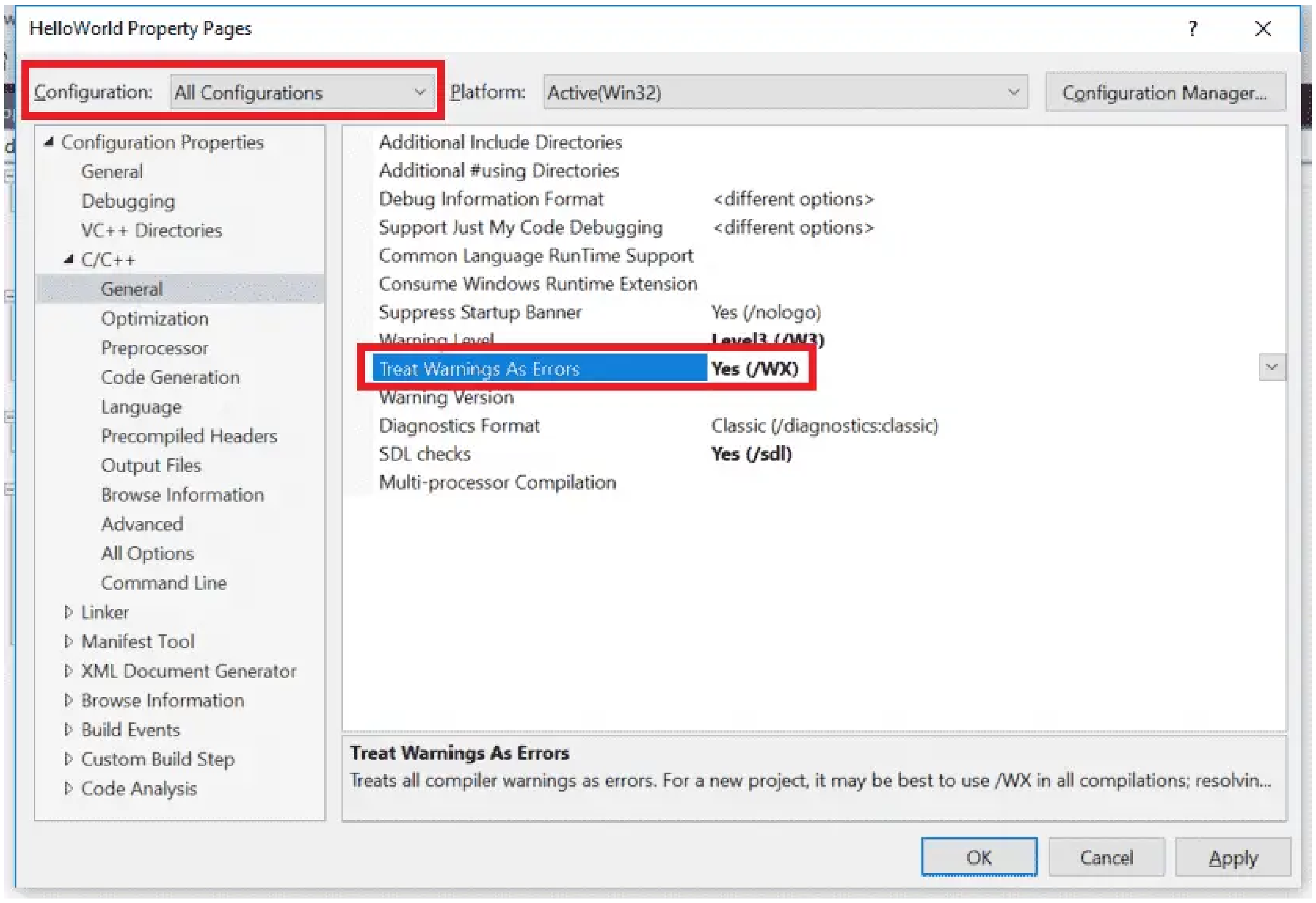
For Visual Studio users

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)*.


HelloWorld Property Pages

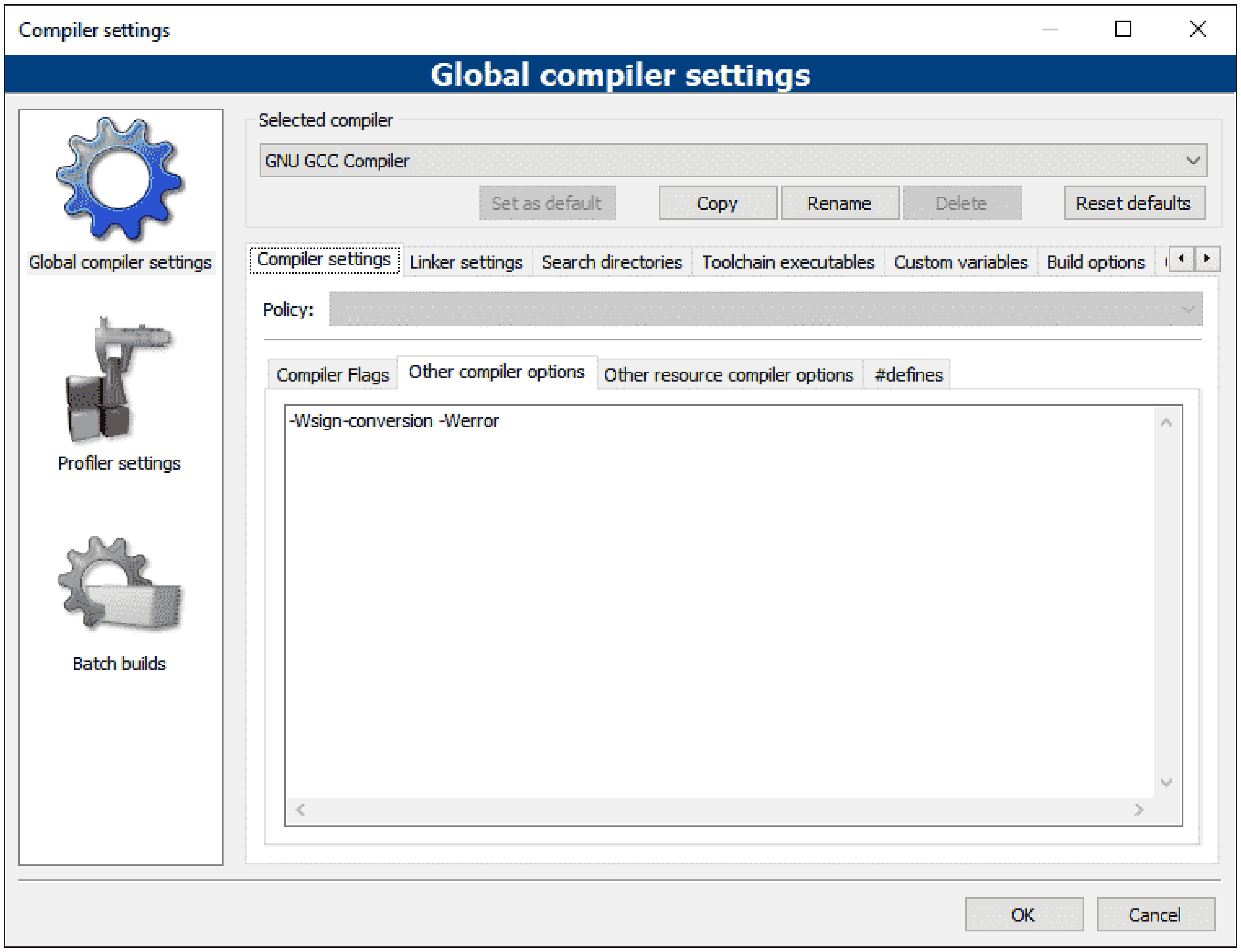
Configuration: All Configurations Platform: Active(Win32) Configuration Manager...

Additional Include Directories	
Additional #using Directories	
Debug Information Format	<different options>
Support Just My Code Debugging	<different options>
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	**Yes (/WX)**
Warning Version	
Diagnostics Format	Classic (/diagnostics:classic)
SDL checks	**Yes (/sdl)**
Multi-processor Compilation	

Treat Warnings As Errors
Treats all compiler warnings as errors. For a new project, it may be best to use /WX in all compilations; resolvin...

For Code::Blocks users

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



```
-Wsign-conversion -Werror
```

For GCC/G++ users

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

For VS Code users

In the `tasks.json` file, add the following flags before "${file}", one per line:

```
"-Werror",
```